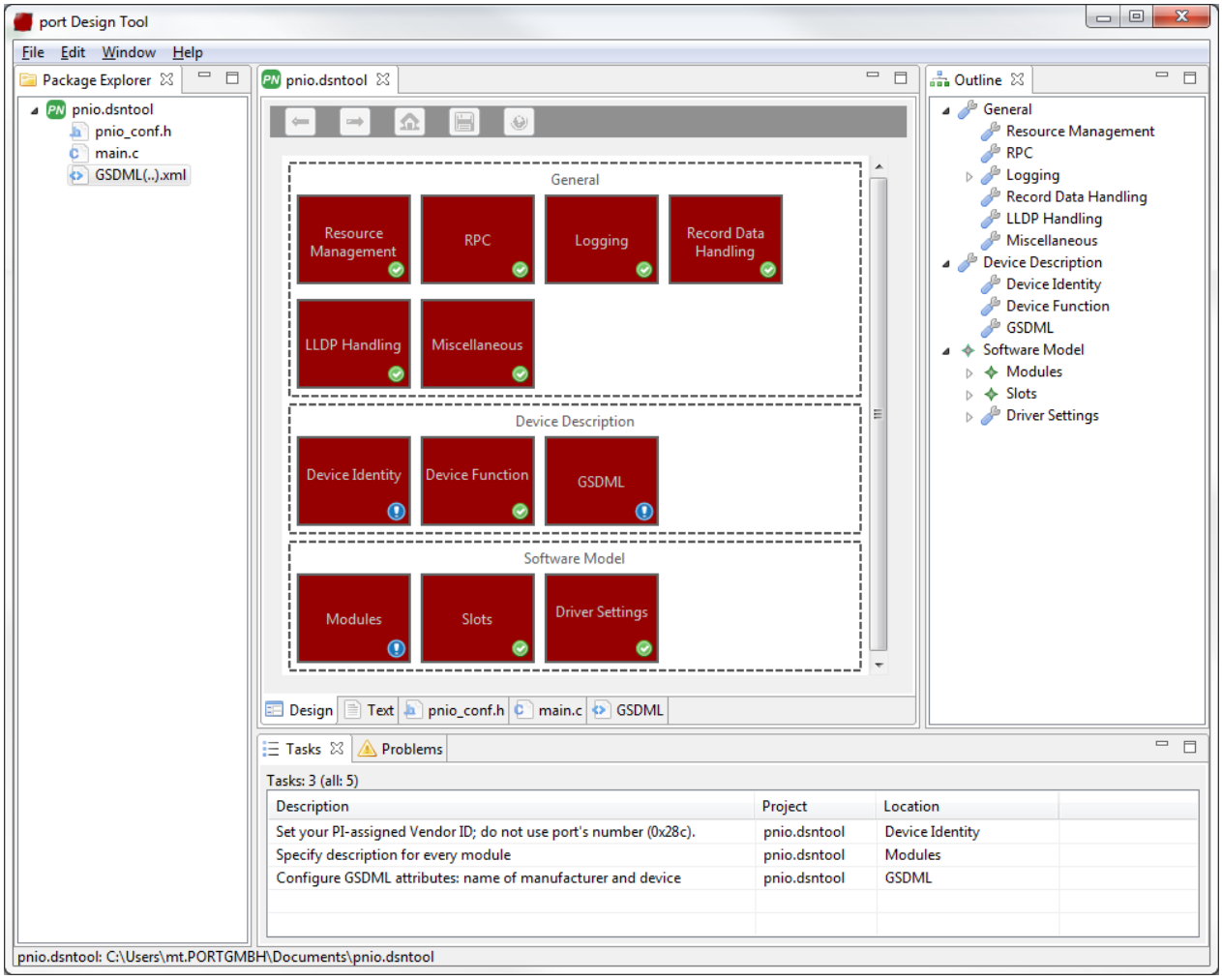
PROFINET DesignTool

User Manual



Disclaimer
All rights reserved

The programs, boards and documentations supplied by *port* GmbH are created with due diligence, checked carefully and tested on several applications.

Nevertheless, *port* GmbH cannot take over no guarantee and no assume del credere liability that the program, the hardware board and the documentation are error-free respective are suitable to serve the special purpose.

In particular performance characteristics and technical data given in this document may not be constituted to be guaranteed product features in any legal sense.
For consequential damages, which are emerged on the strength of use the program and the hardware boards therefore, every legal responsibility or liability is excluded.
port has the right to modify the products described or their documentation at any time without prior warning, as long as these changes are made for reasons of reliability or technical improvement.

All rights of this documentation lie with *port*. The transfer of rights to third parties or duplication of this document in any form, whole or in part, is subject to written approval by *port*. Copies of this document may however be made exclusively for the use of the user and his engineers. The user is thereby responsible that third parties do not obtain access to these copies. The soft- and hardware designations used are mostly registered and are subject to copyright.
We are thankful for hints of possible errors and may ask around for information. We will go all the way to verify such hints fastest.

Copyright

© 2014 port GmbH
Regensburger Straße 7b
D-06132 Halle
Tel. +49 345 - 777 55 0
Fax. +49 345 - 777 55 20
E-Mail: service@port.de
Internet: http://www.port.de

Table of Contents

1 Introduction 9
1.1 Applicability of the Documentation 9
1.2 Product Overview 9
1.3 Product Delivery 9
1.4 System Requirements 9
1.5 Installation 10
2 Introduction to PROFINET 11
2.1 Introduction 11
2.2 The IO Controller 11
2.3 The IO Device 11
2.4 The Supervisor 11
2.5 The IO Data 12
2.6 The IO Record Data 12
2.7 The Alarm Data Objects 12
2.8 Slots and Modules 12
3 Basic concepts 15
3.1 Introduction 15
3.2 Menus 15
3.2.1 The Main Menu 15
3.2.2 The File Menu 16
3.2.3 The Edit Menu 17
3.2.4 The Window Menu 17
3.2.5 The Help Menu 17
3.3 Application Parts 18
3.3.1 Application Perspective 18
3.3.2 Package Explorer 18
3.3.3 Outline 20
3.3.4 Tasks 20
3.3.5 Problems 21
3.3.6 Editor 22
3.4 Editing Tools 27
3.4.1 Configuration widgets 27

3.4.2 Input checking 29
3.4.3 Assistance Options 29
3.4.4 Configuring your project 29
3.5 Projects 29
3.5.1 Project Type 30
3.5.2 Project properties 30
3.5.3 Validation 32
3.6 Tasks 33
3.6.1 Introduction 33
3.6.2 Working with Tasks 34
3.7 Problem Handling 34
3.7.1 Log file 35
3.7.2 Problem types 35
3.7.3 Problem properties 35
3.7.4 Resolving problems 37
3.7.5 Common problems 37
4 PROFINET Design Tool 38
4.1 Introduction 38
4.2 PROFINET Projects 38
4.2.1 Project Type 38
4.2.2 Creating New PROFINET project 39
4.2.3 Specify project file location 39
4.2.4 Specify location for generated code 40
4.2.5 Finish project creation 41
4.2.6 Validation 42
4.3 Editor Extensions 43
4.3.1 Top Page 43
4.3.2 Modules/Slots Page 44
4.3.3 Code Preview 46
4.4 Code Generation 46
4.5 Tasks 48
4.6 Common PROFINET problems 48
4.7 The PROFINET Software Model 48
4.7.1 Modules 49
4.7.2 Submodules 51

4.7.2.1 Name 52
4.7.2.2 Identification Number 52
4.7.2.3 Description 53
4.7.2.4 Data Direction 53
4.7.2.5 Input Data Type 53
4.7.2.6 Input Data Length 53
4.7.2.7 Output Data Type 54
4.7.2.8 Output Data Length 54
4.7.2.9 Subslot Selection 55
4.7.2.10 Parameters 56
4.7.3 Create a new Module 56
4.7.4 Create a new Submodule 58
4.7.5 Slots 60
4.7.6 Subslots 62
4.7.7 Create a new Slot 66
4.7.8 Create a new Subslot 68
4.7.9 Parameters 70
4.7.10 Create/Delete Parameters 72
4.7.11 Settings reference 73
5 Appendix A — Abbreviations 74
6 Appendix B – Common Problems 74
7 Appendix C – PROFINET Task Reference 75
8 Appendix D – Common PROFINET problems 76

List of tables

Table 1: File menu options 16
Table 2: Edit menu options 17
Table 3: Window menu options 17
Table 4: Help menu options 18
Table 5: Project node context menu options 19
Table 6: Outline context menu options 20
Table 7: Task View context menu options 21
Table 8: Problems table context menu options 22
Table 9: Checkbox example 28
Table 10: Numeric setting example 28
Table 11: Hexadecimal setting example 28
Table 12: Text input example 28
Table 13: Drop-down list example 28

List of figures

Figure 1: PROFINET device model 13
Figure 2: Main menu with key selectors 16
Figure 3: Application part label 18
Figure 4: Empty Package Explorer 19
Figure 5: Package Explorer with opened projects 19
Figure 6: Outline View with example contents 20
Figure 7: Tasks View with example contents 21
Figure 8: Problem View with example contents 22
Figure 9: Asterisk on editor label signifies unsaved changes 22
Figure 10: Basic elements of Editor 23
Figure 11: Edit toolbar 23
Figure 12: Top view page for project - an example 24
Figure 13: Editor form page - an example 25
Figure 14: Editor page with tree - an example 26
Figure 15: Elements of code preview tab 27
Figure 16: Error mark displayed next to incorrectly modified intput 29
Figure 17: Mouse-over text contains more descriptive information about the project entry 29
Figure 18: "About" dialog with supported project types 30
Figure 19: Thre projects visible in Packet Explorer 31
Figure 20: Project Properties dialog 31
Figure 21: Tool asks for confirmation if problems were detected during project saves 32
Figure 22: Example contents of Tasks View 33
Figure 23: Task Details dialog for example task 33
Figure 24: Problems View with examples 36
Figure 25: Problem Details dialog 36
Figure 26: Example PROFINET project entry in Package Explorer view 39

Figure 27: Contents of Select Wizard dialog with PROFINET entry 39
Figure 28: First page of New Project wizard 40
Figure 29: Warning that specified file already exists 40
Figure 30: Second page of New Project wizard 41
Figure 31: Warning that a target file already exists 41
Figure 32: Error message when projects files could not be created 42
Figure 33: The user has to confirm file overwrite 42
Figure 34: Dialog with results of explicitly request project validation 43
Figure 35: Top Page for PROFINET project 44
Figure 36: Modules/Slots editor page 45
Figure 37: Tabs for PROFINET editor 46
Figure 38: Block decorations depening on tasks 48
Figure 39: Example device modules and slots 49
Figure 40: Module name input field 50
Figure 41: Module number input field 50
Figure 42: Module description input field 50
Figure 43: Module Slot selection field 50
Figure 44: Warning if slot is already in use 51
Figure 45: Fragment of editor for Module 51
Figure 46: Module's submodule management field 52
Figure 47: Submodule name input 52
Figure 48: Submodule number input 52
Figure 49: Submodule description input 53
Figure 50: Submodule date direction input 53
Figure 51: Submodule input data type 53
Figure 52: Submodule input data length 53
Figure 53: Submodule output data type 54
Figure 54: Submodule output data length 54
Figure 55: Submodule Subslot input 55
Figure 56: Message dialog when selection Subslot fails 55
Figure 57: Dialog warnung that subslot is already in use 55
Figure 58: Submodule parameters input 56
Figure 59: Create Module Wizard 57
Figure 60: Create Module Wizard: Slot selection 58
Figure 61: New Submodule Wizard 59
Figure 62: New Submodule Wizard: Subslot selection 60
Figure 63: Slot number input 60
Figure 64: Slot description input 61
Figure 65: Slot Module input 61
Figure 66: Module already in use warning 62
Figure 67: Fragmen of editor page for Slots 62
Figure 68: Subslots management input 62
Figure 69: Subslot number input 63
Figure 70: Subslot description input 63
Figure 71: Subslot Submodule input 63
Figure 72: Message dialog when selecting Submodule fails 64
Figure 73: Warning when submodule is alread in use 64

Figure 74: Subslot records input 64
Figure 75: Create new Record Wizard 65
Figure 76: Create new Record Wizard: Record Value 66
Figure 77: Create Slow Wizard 67
Figure 78: Create Slot Wizard: Module selection 68
Figure 79: New Subslot Wizard 69
Figure 80: New Subslot Wizard: Submodule selection 70
Figure 81: Paramter name input 70
Figure 82: Parameter index input 71
Figure 83: Parameter description input 71
Figure 84: Parameter data type input 71
Figure 85: Paramater data length input 71
Figure 86: Create new Parameter Wizard 72
Figure 87: Deleting Parameter using Outline View 73

1 Introduction

1.1 Applicability of the Documentation

The documentation of the PROFINET DesignTool by ***port*** consists of a User-Manual and an Online Help System integrated in the DesignTool. Both parts describe the usage of the Unified DesignTool platform and the PROFINET DesignTool in particular.

1.2 Product Overview

For comfortable and easy configuration and object dictionary management, port provides its Unified DesignTool Platform which enables the user to develop communication solutions using port stacks rapidly.

The individual functionality for each stack is realized by plug-ins for each protocol. So the user has only to learn how to work with one tool.

The PROFINET DesignTool contains the Unified DesignTool Platform as well as the plug-in for port's PROFINET IO stack.

1.3 Product Delivery

The PROFINET DesignTool can be downloaded from port's website at http://www.port.de
It comes as a zip or tar archive (depending on the target architecture) which can be extracted using the operating system specific tools.

1.4 System Requirements

The PROFINET DesignTool is available for Linux and Windows operating systems. For both systems versions for 32bit and 64bit architecture are provided. The following windows versions are supported:

- Windows XP
- Windows 7
- Windows 8

Additionally, a Java Runtime Environment (JRE) 7 is required. Depending on the version the DesignTool (32bit or 64bit) that shall be used the according JRE (32bit or 64bit) has to be installed.

Please note, that even if a web browser on the target system is capable of running Java applets, this does not mean that a JRE is installed in the system. To check whether a JRE is installed, open a command interface window, type "java -version" and press enter. If the command could be executed a JRE is installed. Please check the version information whether it reports a JRE 7.

1.5 Installation

To install the DesignTool, extract the provided archive to a directory of your choice. Please ensure that tool has write access to this directory. It is recommended to use a directory within the home directory of your user. There is no further installation required.

You can start the design tool by executing dsntool on Linux or dsntool.exe on Windows.

When running the DesignTool now, it will start in demo mode. Depending on the tool type, some limitations apply.

To get the full functionality, you have to install the license files which can be purchased from port. Please contact service@port.de for further information. To install the license, copy the two file license.dat and license.sig received from port to the directory where you extracted the DesignTool archive. Then restart the tool. It will now provide the full functionality.

2 Introduction to PROFINET

2.1 Introduction

PROFINET is an open industrial Ethernet standard. Its design is utmost modular making it possible to create functionality which is defined by the user. PROFINET IO (PNIO) is real-time capable, so it can be used in almost all fields of automation technology. In a PNIO network you can find three types of devices that interact in a master-slave-relationship:

- IO Controller; master
- IO Device; slave
- IO Supervisor; used for setup and diagnosis

During startup of automation equipment (assuming a complete configuration) an application relationship (AR) is created between a controller and a device. It is a kind of channel where both can communicate with each other. Within an AR three types of communication relationships are usable and have to be defined by the user:

- IO Data CR
- Record Data CR
- Alarms CR

2.2 The IO Controller

The controller has master status in the network. It receives and transmits IO Data from the devices and controls the whole process. The user has to inform the controller about the devices that are in the network by installing a device specific GSD file in the controller. Based on the information that are provided to the controller within this file, the controller automatically creates the AR for every device that is modeled by the user.

2.3 The IO Device

The device has slave status in the network. It receives and transmits IO Data from the controller and provides process information. It is common for device hardware to be to be very modular, which is reflected in the device software model and addressing concept. Within the device model there is a differentiation between addressing and functionality.
A module (or submodule) is hardware with a special functionality and parameters while a slot (or subslot) only represents an address. The module (and submodule) becomes accessible by the controller when it is linked to a slot (and subslot) inside the device software definition, even if the device hardware is not modular. Sometimes it is necessary to submit data in a special structure. If so, different user profiles can be applied which are identified by the Application Process Identifier (API). API 0 is manufacturer specific.
According to this concept a value is addressed inside the device by its API number, slot number, subslot number and in some cases its index.

2.4 The Supervisor

The Supervisor is used to for setting up the controller, e.g. setting IP addresses and installing GSD files, as well as for diagnosis purposes. It is usually not permanently included in the network. If necessary, the supervisor is able to act as controller just as well.

2.5 The IO Data

From the controller's point of view received data is always input data and sent data is always output data. The IO Data CR is used to send and receive process data cyclically. The user can choose different real-time classes that provide diverse options, e.g. isochronous real-time communication.
IO data frames are submitted without handshake due to performance. In case a network member aborts sending IO data to the controller, the application produces an error.
To find out if the submitted data is valid, the process data is followed by a status information inside the submitted frame. IO data are followed by an IO provider status (IOPS) or an IO consumer status (IOCS).
The user has to setup this communication relationship inside the GSD file. Inside the device configuration the adequate IO data objects have to be created that are intended as a kind of link between the process data itself and its corresponding address (api, slot, subslot).

2.6 The IO Record Data

These data objects are sent acyclically and with handshake. Record data objects are used for parameterization of the device or reading parameters from the device. Parameters are usually written by the controller during startup and submitted as record data objects. Inside the device a parameter is addressed by api, slot, subslot and additionally by an index.
The user has to setup this communication relationship also inside the GSD file.

2.7 The Alarm Data Objects

These data objects are sent acyclically and with handshake. They are sent with high priority on occurring events.
There are two main types of alarms:

- Process Alarms
- Diagnosis Alarms

Process alarms are used when a special event in the process occurs. This might be exceeding of a critical value for example. In this case the device itself is still running without errors. Diagnosis alarms in contrast are caused by events or errors corresponding to the IO device. This might be malfunction of attached periphery or pulling and plugging of modules if the device hardware is modular.
Both kinds of alarms can be sent at different priority levels.

2.8 Slots and Modules

If your device hardware is modular (meaning you are able to plug and unplug several modules) you will easily understand the software model. If it is not you have to know that the software model is derived from modular hardware. A module is the interface to your process and it usually has some characteristics and parameters you want to change. This can be done when it is addressable by the controller. A module becomes addressable by the controller when you plug it into a slot, since a slot represents an address. See Figure 1 for a visual representation. Plugging a module means linking the structures of modules and slots

within the software model using pointers.
Figure 1 should only be considered as example. You can create a lot more slots and subslots inside your device.

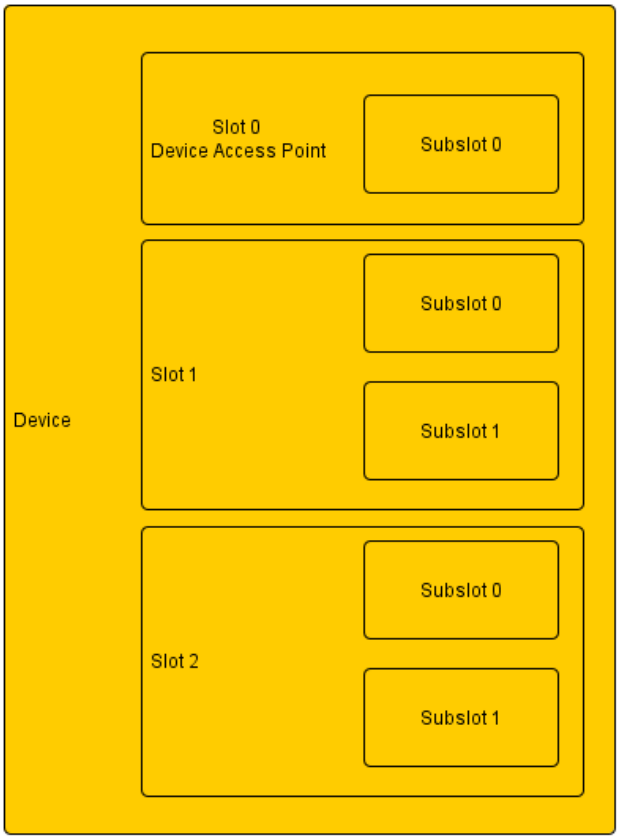


Figure 1: PROFINET device model

Please notice: Slot 0 is pre-defined as Device Access Point (DAP). It represents the device's bus interface and should not be changed.

Subslots, Submodules and IO Data

It is analogue to the slot and module issue, only one addressing level deeper. Within the submodule definition, it has to be declared as Input (from the controller's point of view, the IO device is provider), Output (from the controller's point of view, the IO device is consumer) or none.
Please notice: For a proper configuration, a slot must have at least one subslot defined.

Parameters

Parameters belong to a certain submodule. They are addressed on the submodule by an index. To have them written or read by the controller they have to be linked in the software model to their corresponding Record Data Objects.

General Device Information

There is an amount of general information such as the MAC address, the IP address or the name of the device or the device ID and vendor ID. For a successful ident request, at least the mac address and the name of the device have to be set by the user. Other device parameters like the IP address can be set by a hardware configuration tool once the device is working.

3 Basic concepts

This chapter describes the usage of the DesignTool platform which applies to all available protocol versions.

3.1 Introduction

Using port DesignTool is working with projects. Your DesignTool product can support one or more project types, depending on number of features installed. If in doubt, you can check what project types are supported by opening About dialog from Help menu.

Project configuration is stored in project file on local file system. User is asked to specify location of this file during project creation. For changes not to be lost, project needs to be saved to file after modifications are made.

port DesignTool application window is separated into parts, with movable borders and changeable layout. The placement of application parts is called perspective. Changing perspective can be helpful to adapt tool to screen size and user preferences.

Each opened project can be displayed in its own editor tab located (by default) in center of application window. By interacting with editor, user can browse through project and modify its configuration. Typically configuration settings are laid out in form and use one of few basic input widgets.

If problems are detected by port Design Tool application, it will attempt to collect information about them and present it in user friendly format. User is then able to diagnose the problem and resolve it using this documentation as reference.

Problems may arise if application is unable to access its resources or when project file is corrupted. Problem handling system also covers project type specific checks: correctness of project configuration, setting contradictions etc.

3.2 Menus

3.2.1 The Main Menu

This chapter describes main menu items of *port DesignTool* application. Main menu is positioned on top of *DesignTool's* main window. It is accessible using mouse or keyboard.

When left `Alt` key is pressed, menu control becomes focused and underscores appear on menu labels – see Figure 2. Underscores show which keyboard key to press to select menu item. For example to execute menu action *New*, one can press: `Alt`, `f`, `n`.

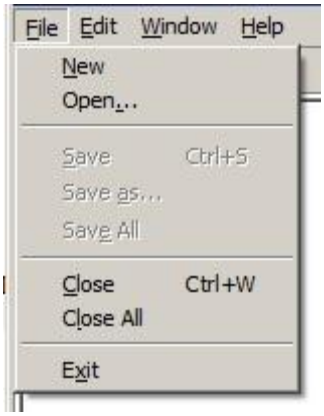


Figure 2: Main menu with key selectors

Some commands available as menu items are also executable using direct keyboard shortcuts. For example, user can undo last modification done in currently focused editor by pressing Ctrl+z. If menu command has key shortcut, it will be displayed next to its menu label.

Not all menu items are available all the time. Depending on application state (active application parts, selected objects, etc.), some items may be grayed-out and not available for selection in given moment.

3.2.2 The File Menu

File menu contains main application actions and basic project management. Short information about each of the items is collected in the table below:

Table 1: File menu options

Item	Description	Shortcut
New	Opens Wizard for creating new projects. If application supports more than one project type, it first queries user which wizard to use. Provided wizard finishes with creation of new project, new editor instance will be opened and brought to top.	
Open...	Opens "select file" dialog and then attempts to load project from selected file. If operation is successful, new editor instance will be opened and brought to top.	
Save	Saves changes done in currently active editor to the project file.	Ctrl+s
Save As	Opens "select file" dialog to obtain new location for project file edited in currently active editor. Can be used to create copy of project file.	
Save All	Saves changes done in all opened editors. Equivalent to executing *Save* on every editor.	
Close	Close project edited in currently active editor.	Ctrl+w
Close All	Close all projects.	
Exit	Close application.	

3.2.3 The Edit Menu

Edit menu contains actions for editing project data. Short information about each of the items is collected in the table below:

Table 2: Edit menu options

Item	Description	Shortcut
Undo	Undo last modification of project data in currently active editor.	Ctrl+z
Redo	Performs again action that was most recently undone.	Ctrl+y
Cut	Cuts text selection from editor or object selection from Outline and puts it into clipboard.	Ctrl+x
Copy	Copies text selection from editor or object selection from Outline and puts it into clipboard.	Ctrl+c
Paste	Pastes outline selection into active editor widget or Outline object.	Ctrl+v

3.2.4 The Window Menu

Window menu contains actions for changing application perspective and to reveal preferences dialog. Short information about each of the items is collected in the table below.

Table 3: Window menu options

Item		Description
Reset perspective		Restores application perspective to default state - this means restoring default positions of application parts and opening closed parts.
Show		Sub-menu for showing selected application part.
	Outline View	Opens Outline part if closed or hidden and brings it to top.
	Package Explorer View	Opens Package Explorer part if closed or hidden and brings it to top.
	Problems View	Opens Problems part if closed or hidden and brings it to top.
	Tasks View	Opens Tasks part if closed or hidden and brings it to top.
Preferences...		Opens dialog for changing application preferences.

3.2.5 The Help Menu

Help menu contains actions related to user assistance and product information. Short information about each of the items is collected in the table below:

Table 4: Help menu options

Item	Description	Shortcut
Help Contents...	Opens help dialog and displays user guide document.	F1
Show Licence...	Opens dialog for displaying text of product licenses.	
Check for Updates	Checks if updates are available for *port DesignTool* application. In case they are, it can download and install them. Requires Internet connection. Update installation requires restart of application.	
About	Displays concise information about *port DesignTool* product including supported project types.	

3.3 Application Parts

This chapter describes the different elements of the DesignTool main window.

3.3.1 Application Perspective

Application window of *port DesignTool* is divided into several application parts:

- Package Explorer view
- Outline view
- Tasks view
- Problems view
- Editor

Layout of these parts in application window is called *perspective*.

Perspective can be changed at runtime for better ergonomics. For example, selected part can be maximized to fill the window contents. This can be achieved by clicking the button next to part label - see Figure 3, ③. Action can be undone by clicking the same button again.

Figure 3: Application part label

Other options for modifying perspective include: hiding (Figure 3, ②) or closing (Figure 3, ①) selected parts. One can also swap positions of parts by drag-and-dropping the part label, or change size of part by dragging part border.

All changes to perspective are persisted when application is closed. To restore default layout, use menu item „Reset Perspective”.

3.3.2 Package Explorer

Package Explorer shows opened projects and their contents. When application is opened with no

projects, the part looks similar to Figure 4. *New* and *Open* are shortcuts for creating new projects and opening existing ones.

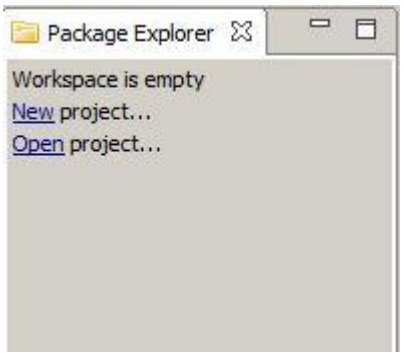


Figure 4: Empty Package Explorer

When one or more projects are opened, Package Explorer displays them as tree nodes - see Figure 5. The icon is specific to project type, with the exception that yellow folder icon ( ) represents project not yet loaded. Depending on project type, some projects can have contents (sub-nodes).

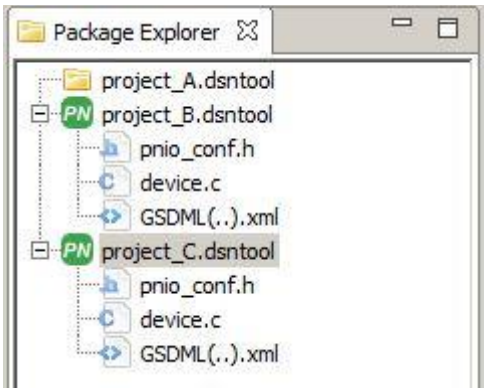


Figure 5: Package Explorer with opened projects

Double click on node will cause project or project content to be opened in the editor. Right click on project node opens context menu. There are following menu items available.

Table 5: Project node context menu options

Item	Description
New	Creates new project using wizard. Equivalent to main menu's New item.
Close Project	Closes selected project. Will attempt to close editor, and then remove project node from Package Explorer tree.
Close Others	Closes all projects except selected one.
Validate	Performs project validation.
Properties	Displays dialog with basic information about project: name, location of project file in the file-system and type of the project.

If Package Explorer is hidden, it can be restored using main menu item Show Package Explorer

View.

3.3.3 Outline

Outline view is, by default, positioned on the right side of application window. This application part presents hierarchy of data opened in currently focused editor. By selecting tree nodes on the Outline part, user can cause relevant page to be revealed in the editor.

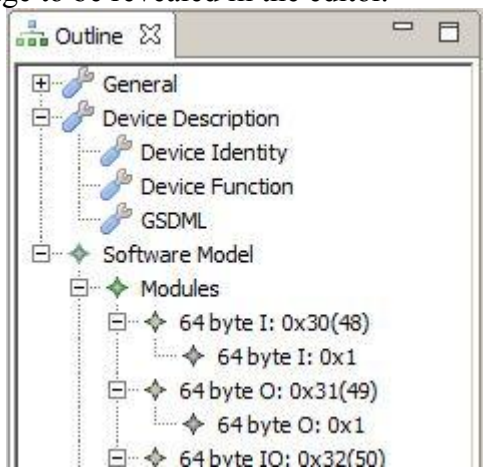


Figure 6: Outline View with example contents

Outline context menu available through right mouse button, allows to perform actions on selected tree node. Actions are not available for every node.

Table 6: Outline context menu options

Item	Description
Cut	Cuts selected object - deletes it from current location and puts it into clip-board.
Copy	Copies selected object to clipboard.
Paste	Adds object from clipboard to children of selected object.
Delete	Deletes selected object.

If Outline view is hidden, it can be restored using main menu item Show Outline View.

3.3.4 Tasks

Tasks view is, by default, positioned on the bottom of application window. This application part contains table with all tasks associated with currently opened projects.

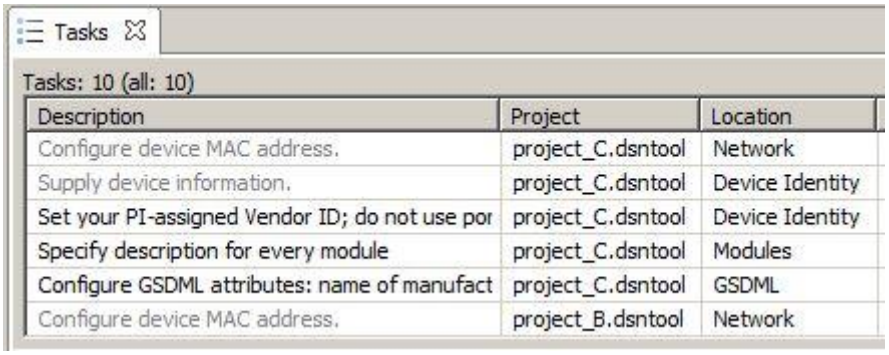


Figure 7: Tasks View with example contents

Task View application part filled with example task entries can be seen on Figure 7. Numbers on top of table refer to numbers of items shown and - in the parenthesis number of all items (including hidden ones). Finished tasks are either hidden (by default) or grayed out.

The table has three columns:

- **Description:** Describes actions necessary to complete task; if too long to read, use context menu to display task details;
- **Project:** Name of project;
- **Location:** Name of root element inside project that needs to be modified for task to be completed.

Context menu, available through right mouse button, has following items:

Table 7: Task View context menu options

Item	Description
Go To	Reveals location associated with selected task in the editor. Equivalent to double clicking on task.
Set As Completed	Temporarily sets selected task as completed.
Show/Hide Completed	Shows or hides completed tasks
Refresh	Checks all tasks and updates their status.
Details	Displays dialog with details of selected task.

If Tasks view is hidden, it can be restored using main menu item Show Tasks View.

3.3.5 Problems

Problems view is, by default positioned on the bottom of application window. This application part contains table with most recent problems identified during application life cycle.

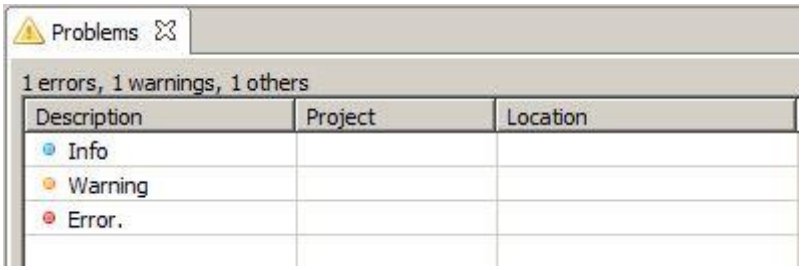


Figure 8: Problem View with example contents

Problems View can be seen on Figure 8. The problems table has three columns:

- **Description:** Information about problem; color of icon depends on severity: red - error, yellow - warning, blue - information;
- **Project:** Name of project, if problem can be linked to single project;
- **Location:** Location inside project where problem was identified;

Context menu, available through right mouse button, has following items:

Table 8: Problems table context menu options

Item	Description
Go To	Reveals location associated with selected problem in the editor. Equivalent to double clicking on problem.
Details	Shows separate dialog with details of the selected problem.

3.3.6 Editor

Editor is the most important application part of *port DesignTool* and the only one that can have multiple instances in one application window - one for each project. The purpose of editor is to present user with convenient tools to modify project.

Tracking changes

Editor keeps track on project data modifications. When changes are made through editor, that are not yet saved to project file, asterisk appears next to editor label (Figure 9).

Figure 9: Asterisk on editor label signifies unsaved changes

Because editor remembers subsequent changes done to data, there is possibility to undo and re-do them.

Editor elements

The way project is presented in the editor differs depending on project type and should be described

in the User Guide for appropriate feature. Nevertheless, basic elements of editor are shared. These elements can be seen on Figure 10.

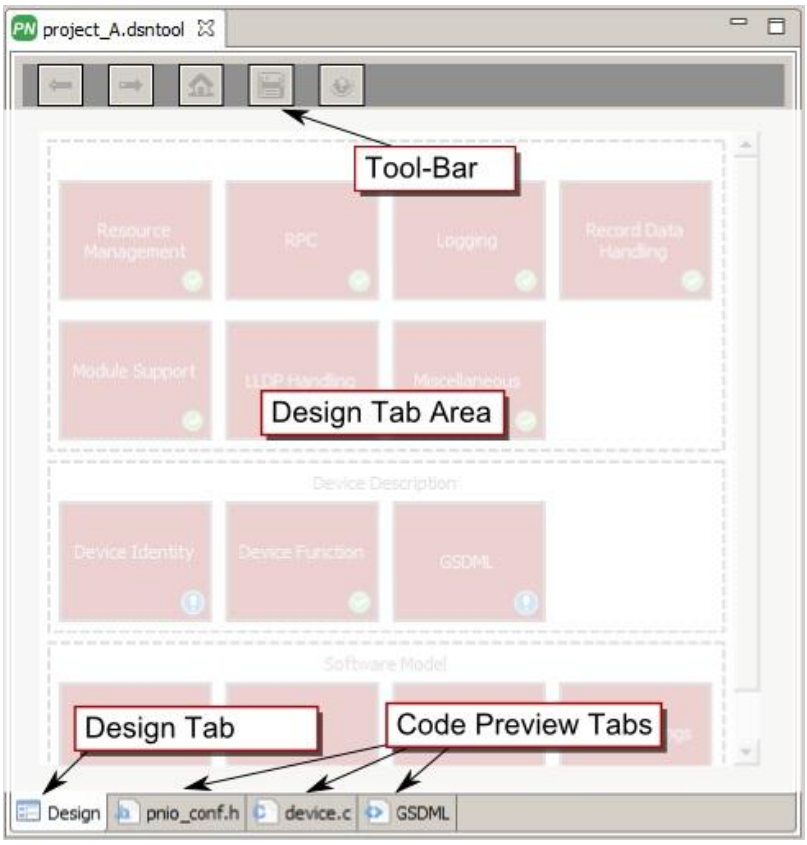


Figure 10: Basic elements of Editor

Editor has multi-tabbed design. *Design Tab* is the main tab, present in every editor. This tab is used to browse and modify the data. Other than that, depending on project type, editors can have additional tabs, for example for viewing generated code.

Design Tab

On top of Design Tab there is a small toolbar mostly for navigation purposes. The rest of tab is taken by editor page - the representation of some aspect of the edited project. Toolbar is presented in Figure 11.

Figure 11: Edit toolbar

The following elements can be found on the toolbar:

- **Go Back:** Opens previously visited editor page;
- **Go Forward:** Opens editor page visited before using *Go Back* tool;
- **Go Home:** Opens top page for given project;
- **Save:** Saves changes to project file; equivalent to main menu's Save item
- **Help:** Opens Help Dialog; if possible uses dialog to reveal help page most appropriate to current editor page.

Editor works in tandem with Outline view. Outline is used to show the hierarchy of project objects and quickly navigate through them - clicking tree-node in Outline results in opening editor page for this object in editor's Design Tab.

There are three basic types of editor pages. Starting page for projects (also known as Top page), uses graphical representation of most important elements of project. Sets of configuration elements are represented as form pages, similar to HTML forms. Hierarchical objects are represented by pages with additional tree for easy navigation through their children.

Top page

Top page for project uses block diagram to present user with key elements of project configuration. This is initial page, displayed after project is created/opened. You can quickly return to this page by using *Go Home* ( ) button on editor toolbar. Exact looks of the page depends on your project structure - defined by project type, but it is similar to that shown on Figure 12.

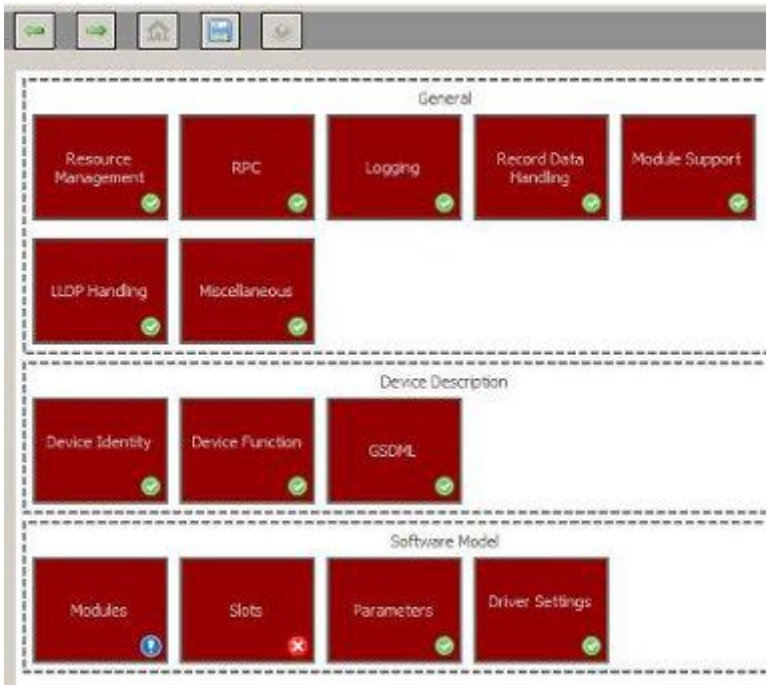


Figure 12: Top view page for project - an example

Blocks (in burgundy-red) are hyper-links to other editor pages. As a visual hint, little decoration icons can be rendered on top of these blocks. Below are their meanings; they are listed in decreasing priority:

- Problems detected for objects inside this block.
- There are unfinished tasks associated with objects inside this block.
- Neither problems, nor unfinished tasks found for this element.

Form page

Using editor's design tab, one can display several configuration pages for project. They can be accessed by clicking a block on Top page, selecting Outline node or double clicking on Task. Remember that you can always go back to previously visited location using *Go Back* (⬅) tool.

Most of these pages use forms, similar to these on WWW pages. See Figure 13 for reference - each configuration setting is displayed with short name on the left and interactive widget on the right.

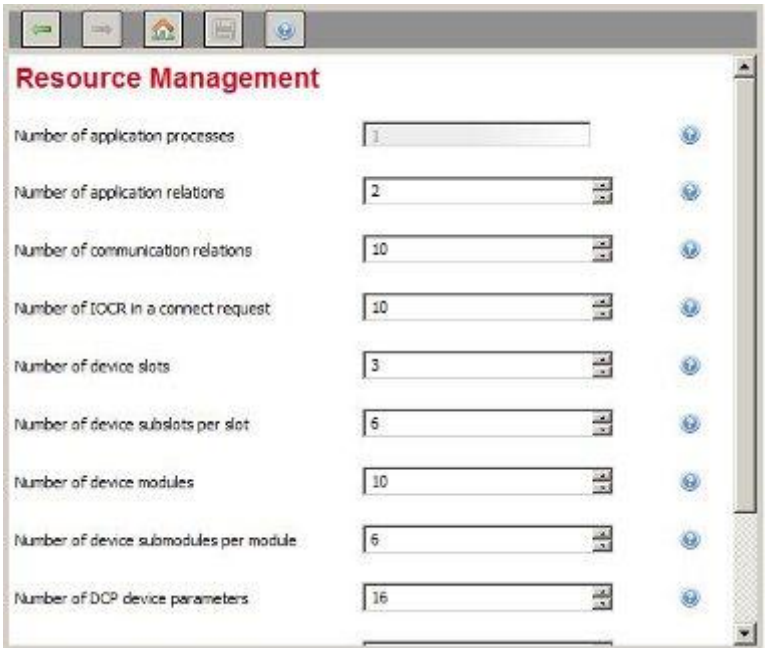


Figure 13: Editor form page - an example

Please see the next chapter for detailed description of commonly used widgets and other editing tips. For specific information about settings and objects available in your project, please refer to user guide appropriate for your project type.

Form page with tree

You can also encounter editor pages that have tree widget nested inside. They are used for objects that have complicated hierarchy and are otherwise not different from ordinary form pages.

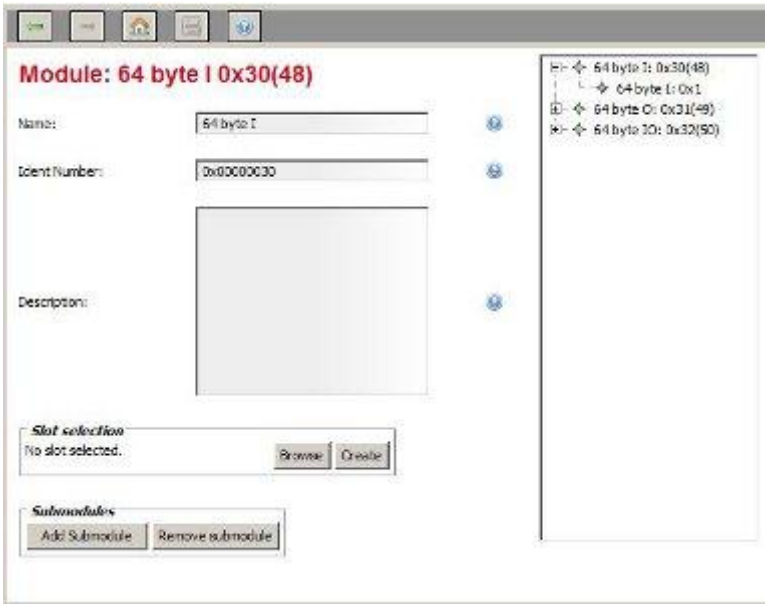


Figure 14: Editor page with tree - an example

Use the tree to access objects hierarchically linked to currently displayed - click on the node to fill editor form with appropriate configuration entries.

Code Preview Tab

Apart from Design Tab, editor instance can have one or more code-preview tabs. You can select one of them by clicking tab label at the bottom of editor area (see Figure 14). Code preview tab allows for quick preview of artifacts generated by *port Design Tool* from currently edited project. They also control location of generated code in file system.

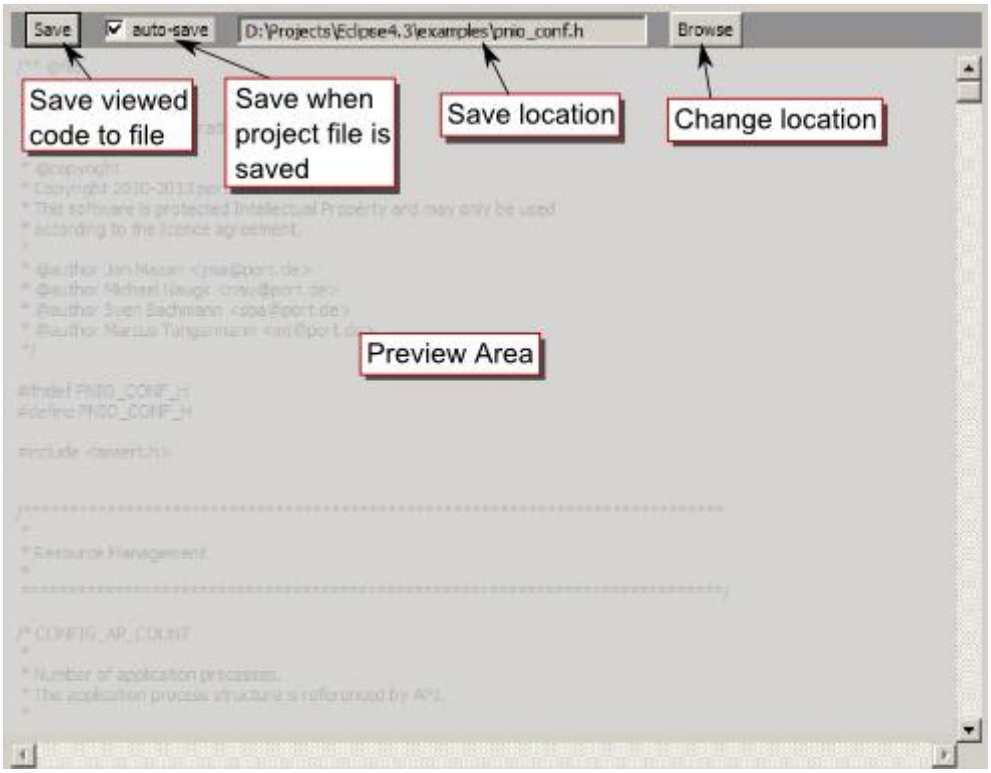


Figure 15: Elements of code preview tab

You can see elements of code preview tab on Figure 15. Descriptions follow below:

- **Preview area:** Most of space in the tab is taken for text field where generated code is displayed.
- **"Save" button:** Saves contents of preview area to file.
- **"auto-save" check box:** If checked, code will be regenerated and saved to file each time parent project is saved. If unchecked, user has to do it manually by clicking "Save" button.
- **Save Location:** Text field in the middle of toolbar displays currently specified location where code should be saved.
- **"Browse" button:** Allows to change location where code should be saved.

Please note: Saving the generated files might be limited in demo mode!

3.4 Editing Tools

This chapter explains in detail types of configuration widgets used for different project settings. For basic information about editor, please see Editor chapter first.

3.4.1 Configuration widgets

When browsing through *port DesignTool* project structure using editor, you will encounter different sets of configuration options, represented by input fields: text fields, lists, check-boxes etc. Each of these widgets was designed to be intuitive in use and to ensure proper format of input.

Table 9: Checkbox example

Example	
Description	Binary (true or false) settings are represented by check-box. Selected means true.

Table 10: Numeric setting example

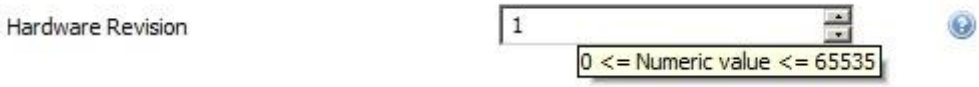
Example	
Description	Numeric settings are represented by "spinner" input. You can change value either clicking on the little up or down arrow buttons or by keyboard. Numeric inputs can be constrained to specific range in which value is valid. Valid range is visible through tool-tip message which pops up when mouse cursor is hovering over the control.

Table 11: Hexadecimal setting example

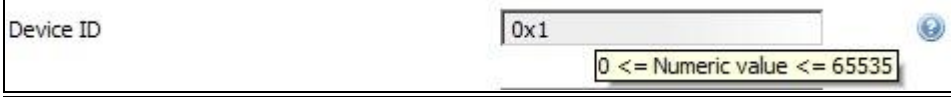
Example	
Description	Sometimes the hexadecimal representation of integer number is more user-friendly than decimal one. For these cases, text field input is used. User can put new value (e.g. *19*) in format he prefers best: • decimal (*19*); • hexadecimal - with *0x* prefix (*0x13*).

Table 12: Text input example

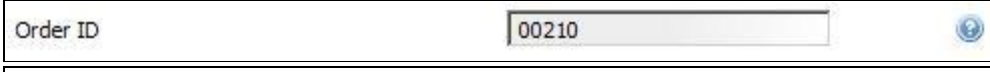
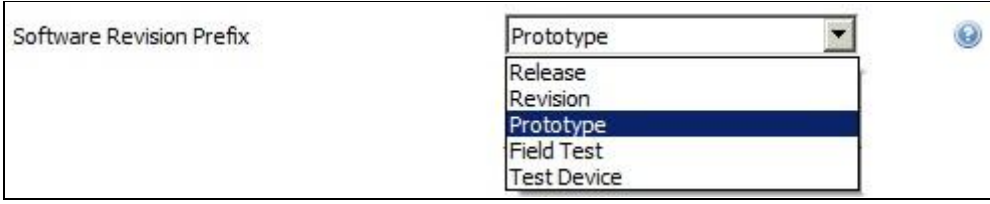
Example	
Description	Settings which expect character string input are represented by text fields. These inputs are often constrained and only accept values that fit special pattern.

Table 13: Drop-down list example

Example	
Description	Settings with discrete set of possible values use drop-down list which allows user to

	select new value from list of possible ones.

3.4.2 Input checking

Before new value is accepted, it must pass quick validation test. For example most numeric settings only accept values which belong to certain range. The test is performed immediately after user modifies input. In case of failure user gets visual feedback - see Figure 16.

Figure 16: Error mark displayed next to incorrectly modified intput

Appearance of error mark next to modified input means that new value is incorrect and not accepted.

3.4.3 Assistance Options

Effort was made to make editing *port DesignTool* projects as intuitive and self-explanatory as possible. Nevertheless, it is still possible that users will not immediately recognize meaning of some setting and how to change its value. For this purpose user assistance system was connected to each configuration entry.

In case the name of the setting is not descriptive enough, hover the mouse cursor over the help icon () next to input in question to get a longer description in form of tool-tip message (Figure 17).

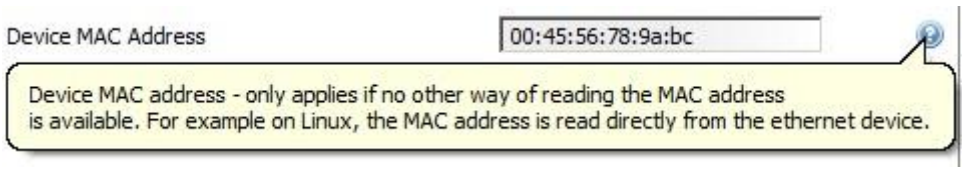

Device MAC Address
00:45:56:78:9a:bc
Device MAC address - only applies if no other way of reading the MAC address
is available. For example on Linux, the MAC address is read directly from the ethernet device.

Figure 17: Mouse-over text contains more descriptive information about the project entry

If this short information is still not sufficient, click on the help icon. This will open user guide for your project type and reveal page most relevant to the setting in question.

3.4.4 Configuring your project

port Design Tool is designed for working with different project types using the same set of convenient tools and user assistance system. The general instructions about editing tools were presented above. For more specific instructions about meaning and significance of settings, refer to user guide appropriate for your project type.

3.5 Projects

3.5.1 Project Type

port DesignTool has a modular architecture, that allows using the same set of tools for editing different project types. You can check what project types are supported by given *DesignTool* application by opening About dialog - see Figure 18.

Figure 18: "About" dialog with supported project types

Support for project type, comes with Wizard for creating new projects. When new project is to be created, user will be asked to choose wizard for the desired project type. The question will be skipped if there is only one wizard available.

3.5.2 Project properties

You can see list of opened projects using Package Explorer. If hidden, restore Package Explorer using menu item “Show Package Explorer”. The content of Package Explorer View includes all projects created and loaded from disk during application runtime, which have not been explicitly closed.

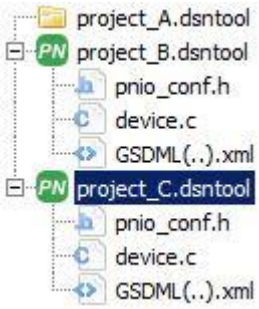


Figure 19: Three projects visible in Packet Explorer

On Figure 19 you can see example contents of Package Explorer View: three project entries. Each project entry is labeled with project name, which is equivalent to its file name. The same project name will be displayed on associated editor label. Icon entry is characteristic to project type.

A yellow folder ( ) is a generic icon used in following circumstances:

- Project is not loaded yet;
- Project type is not supported by the tool;
- Project failed to load (problem with file access or file corrupted);

If application is launched with many projects from previous session, not all of them will be loaded from disk at the same time to save resources.

If project fails to load (second and third case) user can try to diagnose the situation by checking reported problems.

You can open editor for project by double clicking its entry in Package Explorer view.

You can remove project entry from view by right clicking it and selecting *Close Project* from context menu. The same effect can be achieved using main menu. Closing the project doesn't delete its project file, so the project may be opened for editing later. The application will warn about any unsaved changes. It is recommended to close projects that are no longer in active use to save program resources and remove entries associated with project from Tasks and Problems Views.

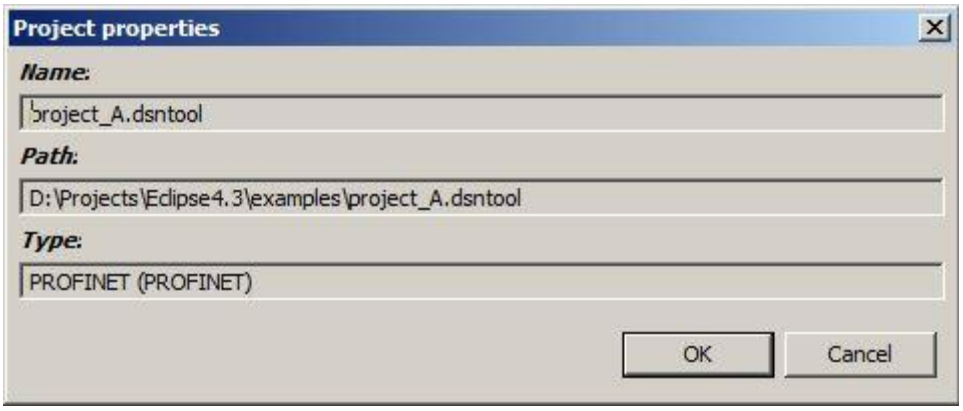


Figure 20: Project Properties dialog

Figure 20 shows an example Project Properties dialog. You can open this dialog by right clicking on selected project entry in Package Explorer and selecting *Properties* from menu. The dialog displays:

- Project name - equivalent to file name;
- Location of project file in file system;
- Type of project.

3.5.3 Validation

Project validation is a process of checking correctness of project data. Validation is supposed to detect problems with project structure that can happen when project file becomes corrupt, as well as smaller issues e.g when two setting values contradict each other.

Project validation is performed in following circumstances:

- On project opening: right after project file is loaded from the disk;
- On project save: right before changes are saved to disk;
- On user request.

Validation may be requested by right clicking project entry in Package Explorer and selecting *Validate* from the menu.

At the start, validation process clears from memory all previously detected problems for project about to be validated. This allows to eliminate from the view, problems that were resolved since previous validation run. After validation completes, you can review all detected problems using Problems View. It can be revealed using Show Problems View menu item.

Automatic validation before project save will interrupt the process, if problems are detected. User will be asked to make decision whether to continue saving invalid project data - see Figure 21. It is recommended to cancel save, check out reported problems, fix them if possible and try again. Users should be aware, that attempting to use such corrupted project to generate code may result in incorrect result.

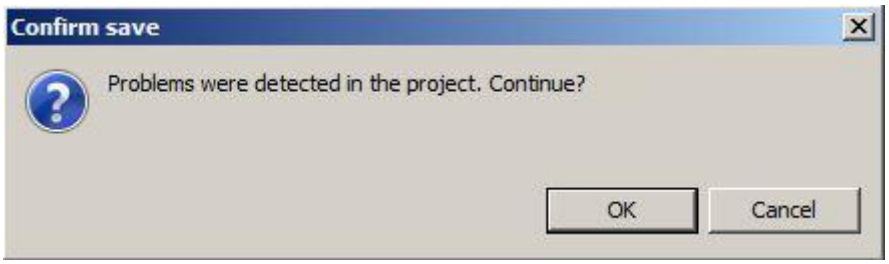


Figure 21: Tool asks for confirmation if problems were detected during project saves

For information about problem handling, and most common problems and solutions, see dedicated chapter. Please, also refer to user guide appropriate to project type.

3.6 Tasks

3.6.1 Introduction

port DesignTool user assistance system comes with task list functionality. Task list is a list of actions that need to be performed to properly configure project. The tool is able to check the project data against the task list and identify which tasks are completed and which remain open. Each project type comes with separate task list.

Tasks application part is used to view all tasks for currently loaded projects. It can be revealed using Show Tasks View menu item.

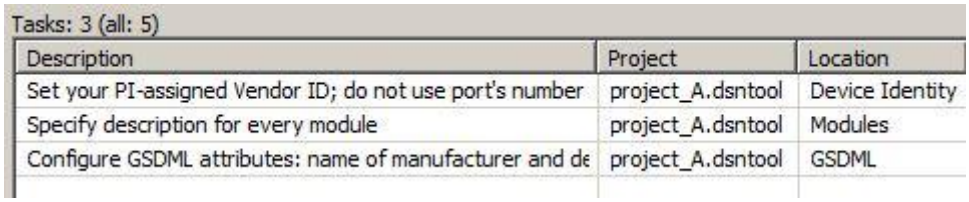


Figure 22: Example contents of Tasks View

On Figure 22, example contents of Tasks View are presented. The entries are grouped by project they apply to. Each task entry can be viewed using separate dialog, which may be more convenient way, especially if task description is long. To display task details right click on task entry and select *Details* from menu.

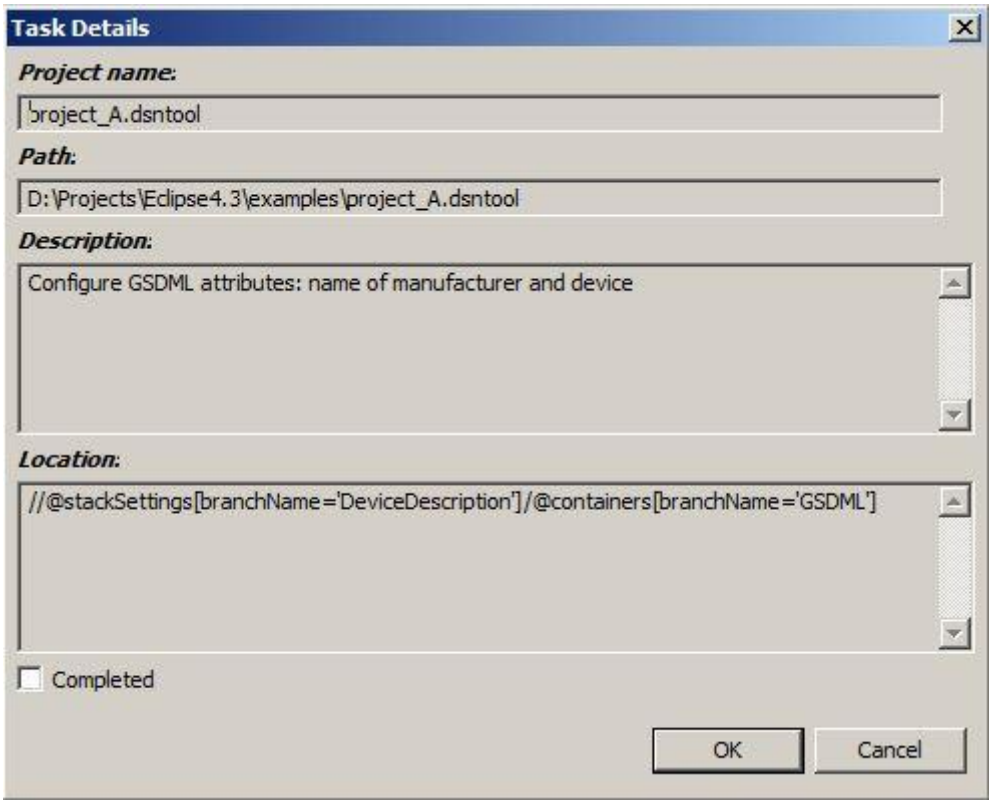


Figure 23: Task Details dialog for example task

Example task opened in Task Details dialog can be seen on Figure 23. Contents include:

- **Project name:** Name of project associated with this task;
- **Path:** Location of project file name in the file system;
- **Description:** Description of task; this field should give information about how to complete task;
- **Location:** Location of object associated with the task inside project; this object will be opened when task entry is double-clicked.
- **Completed:** If the box is checked, task is completed; this can be temporarily changed by the user.

Application checks whether tasks are completed automatically when project is first time loaded and after each save. Checking of tasks can also be triggered manually, by right clicking in Tasks View area and selecting *Refresh* from menu (this checks every displayed task).

3.6.2 Working with Tasks

Reading and completing tasks is a helpful way to properly configure all required aspects of the project. To begin, open Tasks part and scroll to tasks that point to project you are working on. It is usually appropriate to start completing tasks in order they are presented in Tasks View.

First read task description, if description doesn't fit in table view, open details dialog as described above. Double click task entry to open editor page associated with task. When modifications are made, you can re-check tasks using context menu item *Refresh*, or just save project.

If you want to go back to some finished task, you can use context menu item *Show/Hide Completed* to reveal all tasks in the table - completed ones will be grayed out. To help with sorting out the tasks, you can also temporarily change task state to be completed/open by using context menu options *Set Completed* or *Details*. This change will be overridden, the next time task is checked by the application.

If task description is not sufficient in explaining what should be done, refer to user guide for given project type. Double click on the task entry and use editor's help tool, to open help page most relevant to topic at hand.

3.7 Problem Handling

There are different types of problems one can encounter when working with project. Tool may fail to access project file due to Input/Output error, project file may be corrupted and unable to be read or project configuration may be self-contradicting in a way that can be detected using project validation.

In any case, the effort was made to have tool detect most of these errors, and to keep problem reports for user to explore.

3.7.1 Log file

All error handling information is written to log file right after problem is encountered. This file is not generally thought to be of assistance for normal users due to technical nature of its contents. It may be helpful though, to prepare copy of this file when reporting a bug to *port GmbH* user support.

Log file is a normal text file located under path:

{INSTALLATION_DIR}/workspace/.metadata/.log

where {INSTALLATION_DIR} is an address of directory *port DesignTool* was installed into.

3.7.2 Problem types

Problems can be classified depending on their scope and severity.

Problem scope:

- **Project specific:** These problems are associated with concrete project. That includes errors in saving project file or generating code, problems with project configuration, missing values, etc.
- **Global:** These are problems that concern application itself. That includes errors in accessing licence file or auto-update repository, errors caused by corrupted application files, etc.

Problem severity:

- **Errors:** Most severe. Reported when due to a problem, requested operation could not be completed. In case of project problems, error means that project is incorrectly configured and should not be used to generate code.
- **Warnings:** Moderately severe. Reported when requested operation was able to continue, but with complications. Warning reports should be reviewed carefully to check if they do not indicate some larger problem.
- **Infos:** Least severe. Reported for situations that are not likely to be of concern.

3.7.3 Problem properties

Problem reports can be viewed using Problems View. If hidden, it can be revealed using menu item Show Problems View. Figure 24 presents contents of Problems View - the table is filled with example problems of different severity and scope.

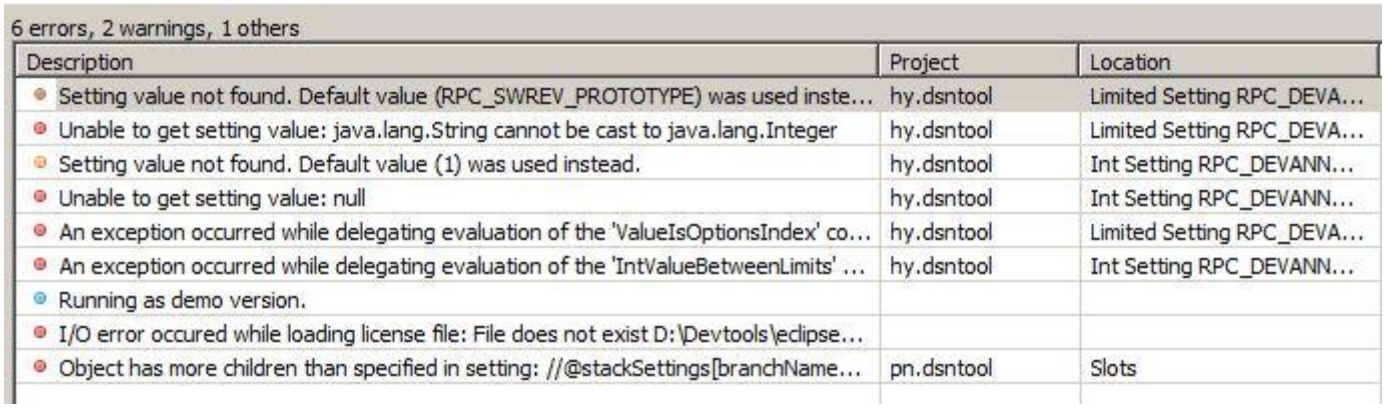

6 errors, 2 warnings, 1 others
Description
Project
Location
Setting value not found. Default value (RPC_SWREV_PROTOTYPE) was used inste...
hy.dsntool
Limited Setting RPC_DEVA...
Unable to get setting value: java.lang.String cannot be cast to java.lang.Integer
hy.dsntool
Limited Setting RPC_DEVA...
Setting value not found. Default value (1) was used instead.
hy.dsntool
Int Setting RPC_DEVANN...
Unable to get setting value: null
hy.dsntool
Int Setting RPC_DEVANN...
An exception occurred while delegating evaluation of the 'ValueIsOptionsIndex' co...
hy.dsntool
Limited Setting RPC_DEVA...
An exception occurred while delegating evaluation of the 'IntValueBetweenLimits' ...
hy.dsntool
Int Setting RPC_DEVANN...
Running as demo version.
I/O error occured while loading license file: File does not exist D:\Devtools\eclipse...
Object has more children than specified in setting: //@stackSettings[branchName...
pn.dsntool
Slots

Figure 24: Problems View with examples

To better examine selected problem entry, user can open separate dialog with problem details. This can be done by clicking on problem entry in Problems View with right mouse button and selecting *Details* menu item. Problem-Details dialog opened for example problem entry can be seen on Figure 25.

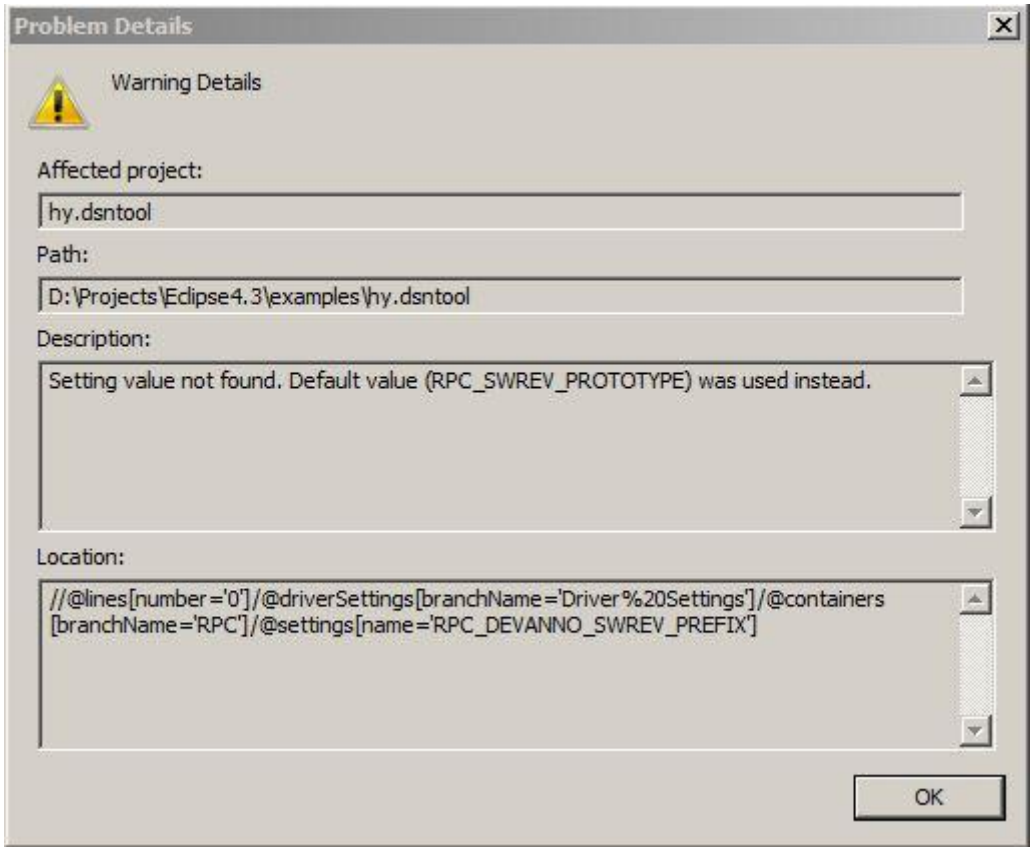


Figure 25: Problem Details dialog

Contents of Problem Details dialog, from the top:

- **Severity:** Symbol (depends on your operating system) and label in dialog header indicate problem severity (Error, Warning or Information);
- **Affected Project:** Only for project-specific problems; name of project in which problem

was detected;
- **Path:** Only for project-specific problems; file-system location of project in which problem was detected;
- **Description:** Description of the problem;
- **Location:** Only for project-specific problems; object inside of project for which problem was detected;

3.7.4 Resolving problems

When problems are indicated (or suspected) in project user is working on, there is a chance they can be quickly resolved using assistance from Problems View and Help documentation.

To browse problem reports open Problems View.

It may be helpful to (re)run project validation. This will cause all currently inactive problems associated with project to be removed from view, and problems reported during validation will appear on top of the list.

Open details dialog for reported problems to read description in full. If report contains specific object where problem is located (Location field is not empty), you can try to open it in editor. To do this, double click on problem entry in Problems View.

If solution is not immediately apparent, check if similar problem is described in the next section. If problematic object or setting is identified, use help tool built into editor ( ) to open help page about it. You may also check, if user guide document for your project type has "Common Problems" section.

After modifications were made to resolve issues, you can re-run project validation and whatever action that had caused problem messages to appear in the first place (e.g. code generation). If problem reports won't appear again, the issue was fixed.

3.7.5 Common problems

For a list of common problems and their solutions please see chapter 6.

4 PROFINET Design Tool

This chapter explains the usage of the PROFINET specific features of the DesignTool.

4.1 Introduction

To start using *port* PNIO stack with own application one has to:

- Define Software Model;
- Create Application Code;
- Create Device Description (GSD/GSDML file);

To assist its customers with getting started quickly, *port* provides *DesignTool* utility application. *port DesignTool* for PROFINET comes with easy to use forms and wizards, which make it easier to configure PNIO stack to user needs.

Functionality provided by the tool include

- Generation of stack configuration file;
- Generation of software model definition file;
- Generation of device description GSDML file;
- User assistance for each configuration setting;
- Detection of common configuration errors;
- Step by step configuration guide in form of task list.

4.2 PROFINET Projects

port PNIO stack configuration, together with software model, is defined as a single *DesignTool* project. Using this project, the tool can generate appropriate code files. Working with PROFINET projects is analogous to working with other *port DesignTool* projects. If some aspects are not clear, please check help pages for common functionality: *port DesignTool User Guide*.

4.2.1 Project Type

User will be able to create and edit PROFINET projects if he has support for this protocol installed into *port DesignTool* application. One can check if that is the case, by opening About Dialog from *Help* main menu and checking if section *Supported project types* contains line "PROFINET".

In Package Explorer View, PROFINET projects are shown with green *PN* icon (PN). Project node contains up to three project contents, as shown on Figure 26. Meaning of these sub-nodes will be explained in Code Generation chapter.

Figure 26: Example PROFINET project entry in Package Explorer view

4.2.2 Creating New PROFINET project

The creation of new PROFINET project is straightforward. From *File* main menu, select *New* option. In case the *DesignTool* product supports more project types, the user will be asked to choose what kind of project to create. Select entry *PROFINET Project (PROFINET)* - see Figure 27 and click *OK*.

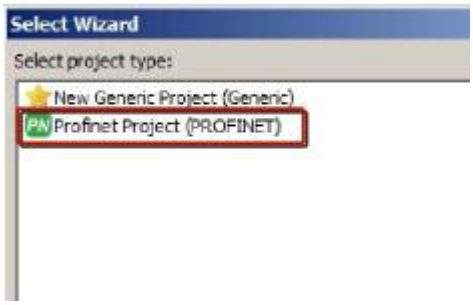


Figure 27: Contents of Select Wizard dialog with PROFINET entry

4.2.3 Specify project file location

On the first page of wizard, user is asked to specify location of project file. Wizard window looks similar to that presented on Figure 28. To proceed, click *Browse* button to display your OS-specific select-file dialog. Project file name has extension *.dsntool*.

Tip: You can organize project files on disk however you want, but consider creating a new folder for each new project. This way it will be easier to know which code file belongs to which project.

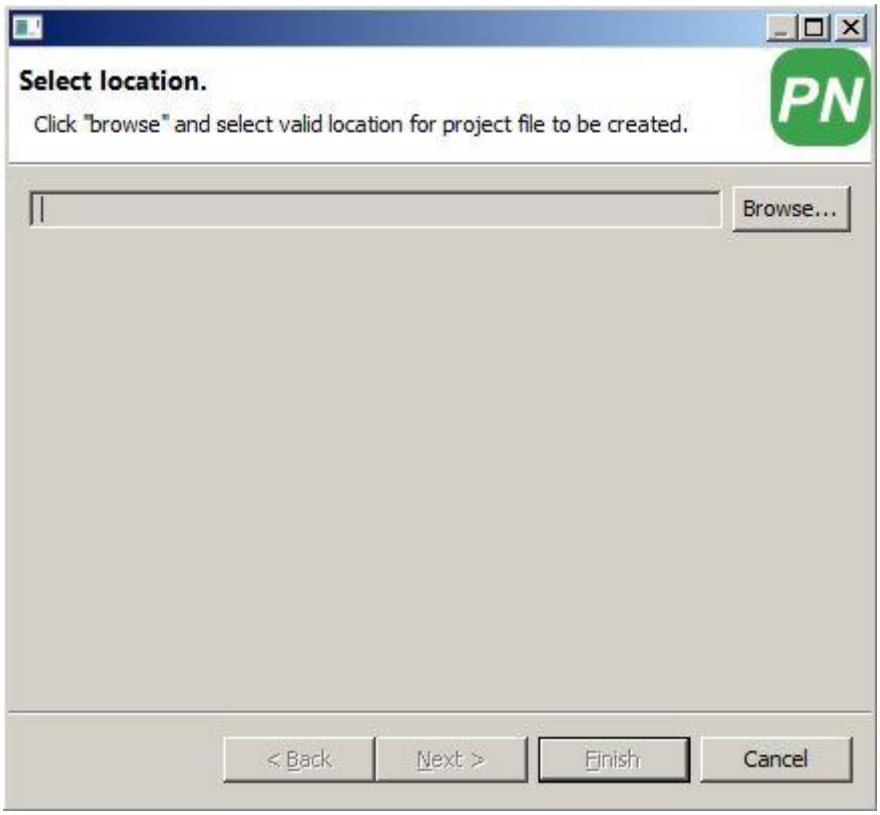


Figure 28: First page of New Project wizard

It is possible to select already existing project file. User must be careful though, as it will result in losing all information saved to this file and replacing it with clean new project. This will happen as soon as Wizard finishes. Wizard warns about overwriting existing files with message shown on Figure 29.

Figure 29: Warning that specified file already exists

After location of project file is specified, proceed to next page by clicking *Next* button.

4.2.4 Specify location for generated code

Second wizard page asks user to specify location of three files that will be generated based on project configuration.

- Stack configuration file *pnio_conf.h*;
- Software model configuration file *device.c*;
- GSDML device description file.

The page is presented on Figure 30. File name of GSDML file is not immediately known as it depends on configuration options. As initial setup, wizard will suggest to locate these files in the same directory as project file.

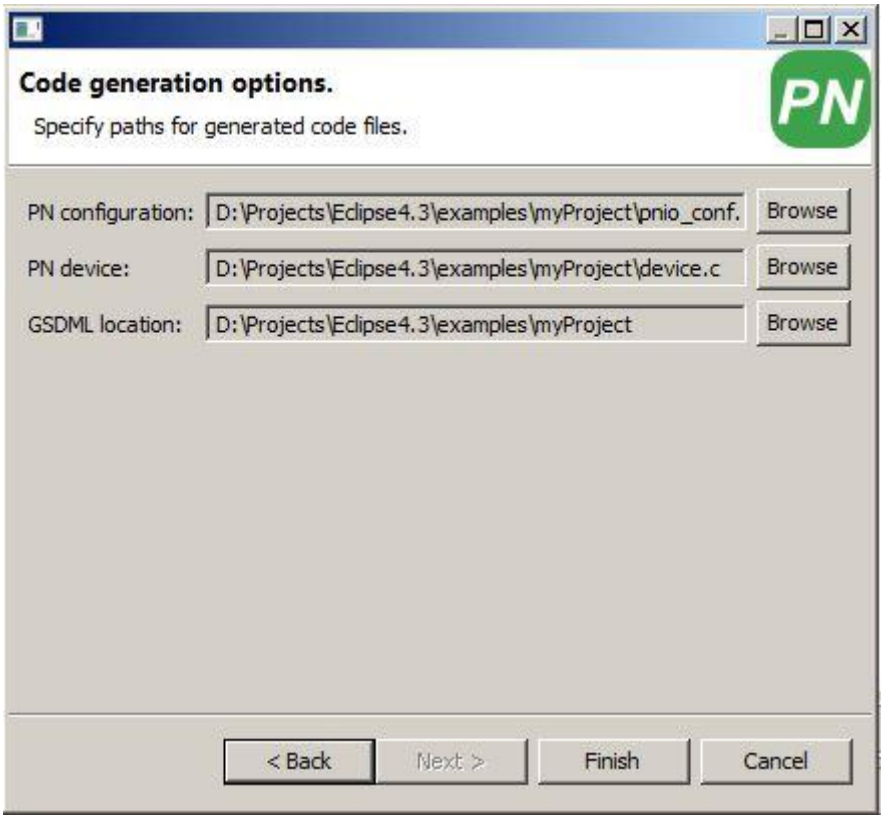


Figure 30: Second page of New Project wizard

You may change location where each of these files is to be saved by using *Browse* button. Be careful to not overwrite some previously created files. Wizard will warn user if *pnio_conf.h* or *device.c* files already exist in location specified - see Figure 31.

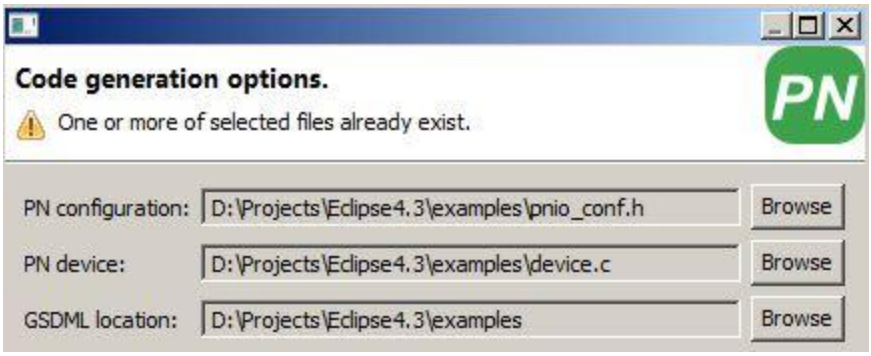


Figure 31: Warning that a target file already exists

4.2.5 Finish project creation

After all locations are set, click *Finish* button to create the project. You may also get back to previous page using *Back* button. If you do not want to continue with new project creation, click *Cancel*.

It may be possible that one of selected locations will be inaccessible for the tool. For example, user running application may not have write rights to selected directory. In this case wizard will not finish, but display error message - Figure 32. In this case user has to change specified path and try again.

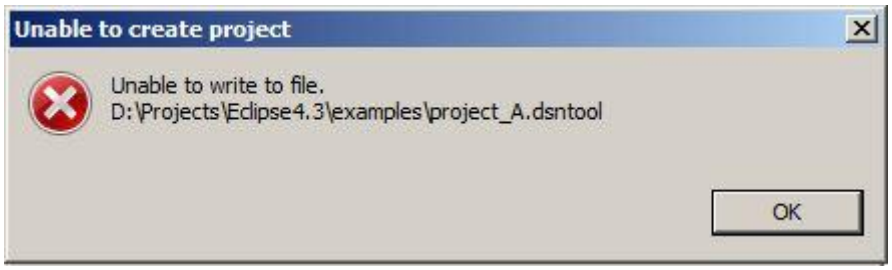


Figure 32: Error message when projects files could not be created

During wizard finish, user will be once again warned about every overwrite risk that was detected, and asked for confirmation - Figure 33. In case of any doubts, click *Cancel* to postpone project creation. Otherwise, click *OK* to continue.

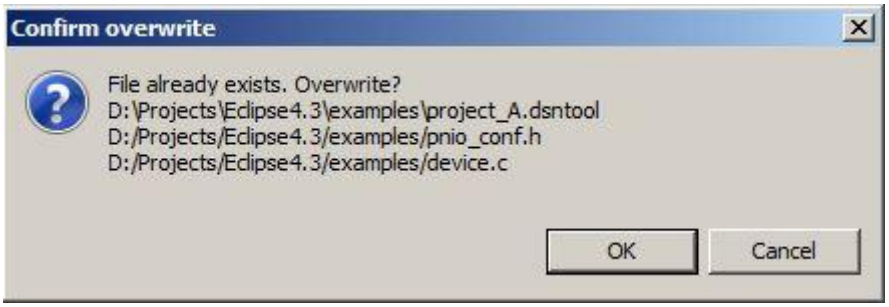


Figure 33: The user has to confirm file overwrite

After project creation wizard finishes successfully, project file will be created and filled with empty project configuration. Other files (*pnio_conf.h*, *device.c*, GSDML) will be created later. The new project will be opened in new editor tab that should appear on top of application window.

4.2.6 Validation

PROFINET projects are subject to correctness checks as every other *port DesignTool* project:

- After project is opened;
- Before project is saved;
- At user request.

You can request project validation using Package Explorer. Right click on project entry in Package Explorer View and select *Validate* menu option. Validation may take a few seconds. After it finishes, dialog with results should appear. If no problems are found, the message will simply say "OK". Otherwise, quick overview of detected problems will be displayed in dialog similar to presented on Figure 34.

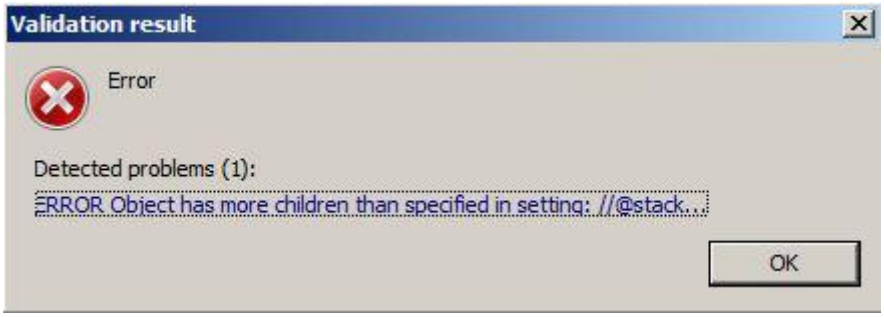


Figure 34: Dialog with results of explicitly request project validation

See Common Problems chapter for list of common issues that may be reported during validation, and how they may be resolved.

PROFINET projects share editing tools with other *port DesignTool* project types. To learn about basic editor elements such as toolbar or input forms, see help pages for common functionality: *port DesignTool User Guide*.

4.3 Editor Extensions

4.3.1 Top Page

After project is loaded, editor will display graphical top page for it. User can always return to this editor page by using *Go Home*tool ( ).

Top page for PROFINET project is presented onFigure 35. Each burgundy-red block is a click-able link to editor page with configuration elements. You can also hover mouse cursor over the figure to display longer description of block contents as tool-tip.

Blocks are organized into three groups:

General
: General Settings of PNIO stack configuration.

Device Description
: Settings related to creation of device description file.

Software Model
: Structure of objects of which application's software model is made plus configuration of the driver.

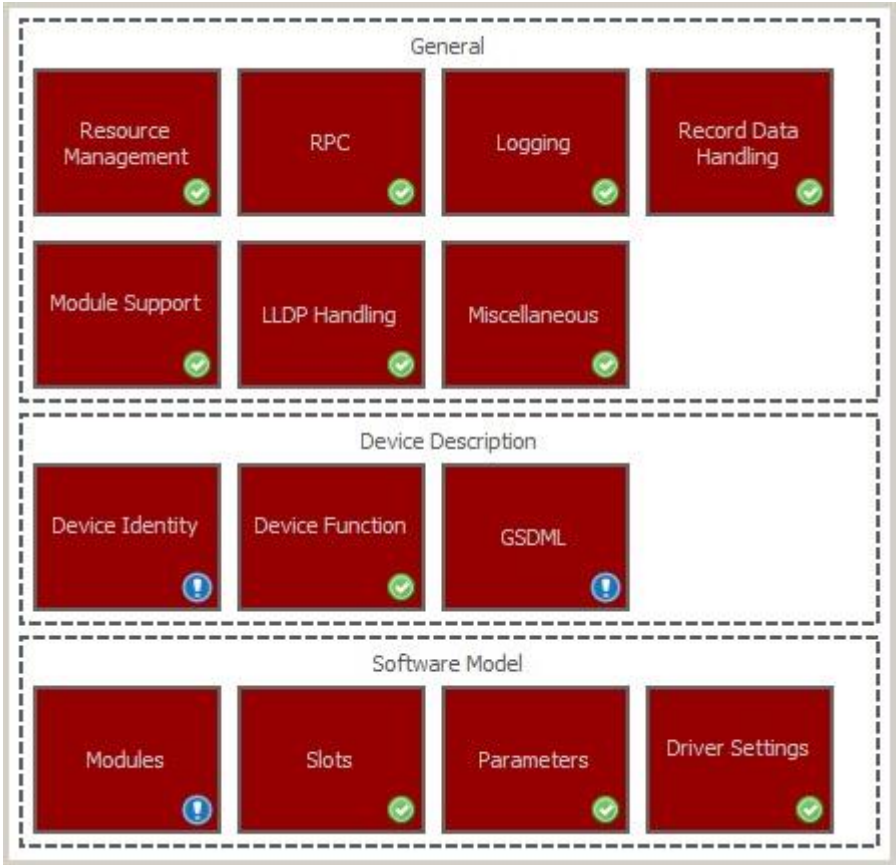


Figure 35: Top Page for PROFINET project

4.3.2 Modules/Slots Page

Modularity of PNIO stack is represented in software model. Process interfaces are separated into Modules and Submodules, while addresses are represented as Slots/Sub-slots. Number and properties of these objects are to be specified by the user, with each new project being initialized with few example Modules and Slots as starting point.

Editor page for managing Modules and Slots is available from top page under blocks *Modules*, *Slots*. The page itself is presented on Figure 36. Be aware that number and names of blocks depend on your current project configuration.

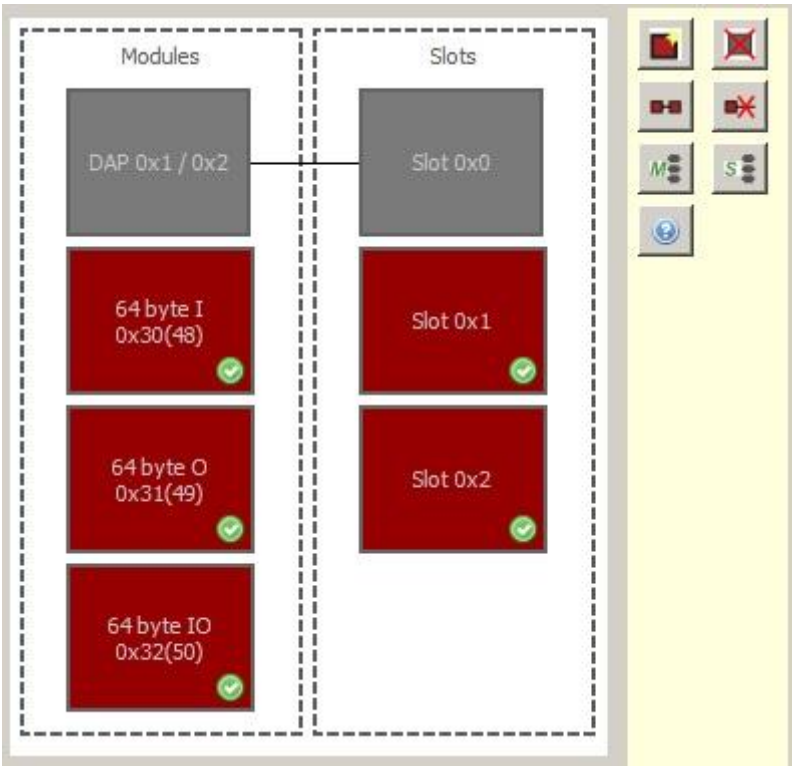


Figure 36: Modules/Slots editor page

Editor page displays diagram of currently specified Modules and Slots. Modules being on the left and Slots on the right, or vice versa - depending what block has led to the page.

Two gray, connected with each other, blocks at top of the page, represent Device Access Point Modules (number 1, 2) and Slot (number 0). These objects cannot be changed, and their identification numbers are reserved - not available for user objects.

Using the page, user can navigate to configuration of specific module/slot (just click appropriate block), add/remove objects and create connections between them. There is an additional, vertical toolbar located on the right side of diagram that contains tools for this purpose.

Add Object

Creates a new Module or Slot. To use, first click the tool button, then click on empty area inside desired group of figures: Modules to create Module, Slots to create Slot.

Remove Object

Deletes Module or Slot and removes it from diagram. To use, first click the tool, then object you want to remove.

Create connection

Plugs Module into a Slot. To use, first click the tool button, then the two objects you want to connect. If you want to connect objects that already have connections, first use *Remove connection* tool.

Remove connection

Un-plugs a Module from Slot. To use, first click the tool button, then one of connected objects.

Align Modules
When clicked, redraws diagram to align each connected Module to its Slot.

Align Slots
When clicked, redraws diagram to align each connected Slot to its Module.

Help Tool
Displays help for the figure. To use, first click tool button, then one of blocks.

4.3.3 Code Preview

Editor for PROFINET projects has three additional tabs beside main Design Tab. All of these are used for preview of generated code.

Figure 37: Tabs for PROFINET editor

pnio_conf.h
Preview of C header file for PNIO stack configuration.

device.c
Preview of C file where software model is defined for use in PNIO stack.

GSDML
Preview of GSDML file displayed using HTML style.

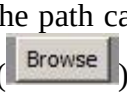
Each time code preview tab is opened, its contents are refreshed to match current project configuration. If project contains errors or inaccuracies previewed code may be incorrect or code will fail to generate altogether. In any case, project validation should provide clues about nature of the problem.

4.4 Code Generation

The PROFINET DesignTool generates several files to be used within the port PROFINET IO Stack. This chapter gives a short overview over the meaning of the different generated files and how to view them.

pnio_conf.h

pnio_conf.h file is a C header file which contains macro-definitions needed for PNIO stack configuration.

Code can be displayed for inspection using appropriate editor tab. Location where *pnio_conf.h* file should be saved is specified when project is created. The path can be seen on top of code-preview tab. User can change this location using *Browse* button (

) in code-preview tab toolbar.

If auto-save check box is selected, the code file will be rewritten, each time project changes are saved to disk. User can also save file manually using *Save* toolbar button.

Contents of *pnio_conf.h* file depend on configuration of project settings. Project settings, their meaning and significance, are described in Settings Reference chapter.

device.c

device.c file is a C code file which contains structure definitions of application software model.

Code can be displayed for inspection using appropriate editor tab. Location where *device.c* file should be saved is specified when project is created. The path can be seen on top of code-preview tab. User can change this location using *Browse* button (

) in code-preview tab toolbar.

If auto-save check box is selected, the code file will be rewritten, each time project changes are saved to disk. User can also save file manually using *Save* toolbar button.

Contents of *device.c* file depend on what Modules, Slots and Parameters are defined. You can get more information about software model in Software Model chapter.

GSDML

Device description, which is needed for IO controller to properly initialize connection with PNIO device, is saved in GSDML format. GSDML is a XML document which contains PROFINET device description (GSD).

Code can be displayed for inspection using appropriate editor tab.

Directory where GSDML file should be saved is specified when project is created. The path can be seen on top of code-preview tab. User can change this location using *Browse* button (

) in code-preview tab toolbar. The file name is created based on GSDML specification (v.2.3) and is made of following segments:

- "GSDML";
- GSDML version;
- Manufacturer Name;
- Device Name;
- Date.

File name segments are configurable.

Tip: It is easy to get confused when changing of GSDML settings cause multiple copies of GSDML files to be created. Therefore it is good practice to keep all files associated with one project in separate folder. Be careful to not incidentally overwrite a GSDML file you have created with other project.

For device description to be displayed in user-friendly format, XSL style-sheet file will be copied into the same directory as GSDML file. This is usually done, first time the file is opened in the pre-

view tab. It may be worth noting, that with style-sheet in place, one can also open GSDML file using other WWW browser.

Contents of GSDML file depend on Software Model and Device Description Settings.

4.5 Tasks

port Design Tool makes use of task lists as a method of assisting users through project creation. Each task is constructed from a set of conditions that are checked automatically each time project is saved. User can browse unfinished tasks using Tasks View application part.

As a means of visual feedback, editor blocks are decorated when there are unfinished tasks detected for their contents - see Figure 38.

Figure 38: Block decorations depending on tasks

For a list of possible open PROFINET specific task, please see chapter 7.

4.6 Common PROFINET problems

During Project Validation problems can be detected in user project. These are collected by Problems View application part, for the user to browse.

In chapter 8 the list of common problem reports concerning PROFINET projects, their meanings and solutions can be found.

4.7 The PROFINET Software Model

Obligatory step of PNIO stack configuration is definition of software model. The software model is supposed to reflect modular and hierarchical structure of hardware it was derived from.

Modules and Slots

The model isolates process interfaces from addressing. The former are represented by Modules, and the latter by Slots. Each of these object types is further separated into one or more child objects. Modules contain Submodules, Slots contain Subslots. On Figure 39 the concept was illustrated with a simple example.

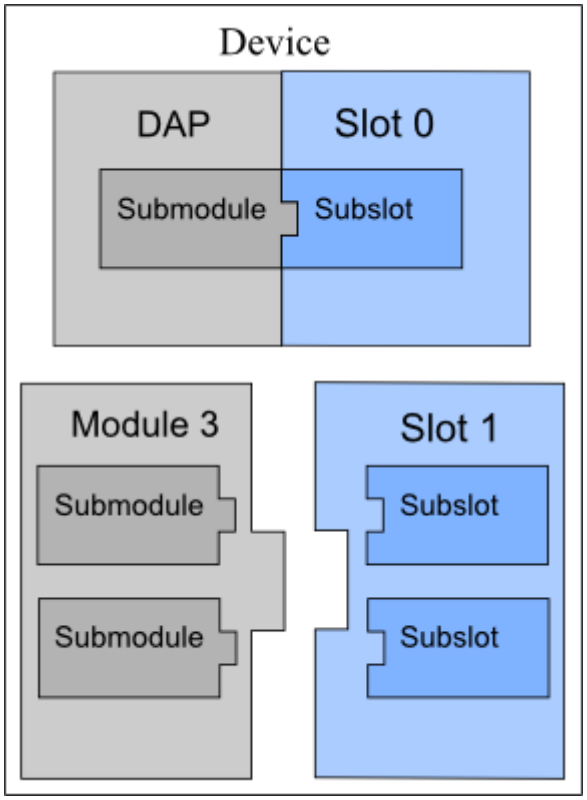


Figure 39: Example device modules and slots

port PNIO device stack comes with predefined Modules (number 1,2) and Slot (0) for Device Access Point (DAP). These objects are not available for editing.

User is free to add and remove objects in software model with the limitation that minimal user-specified software model has one Module and Slot. Also, each Module and Slot must contain at least one child (Submodule, Sub-slot).

Software model can define connection between (Sub)Module and (Sub)Slot (in other words plug Module into Slot).

Parameters and Record Data

Each Submodule object can be parameterized by referencing one or more Parameter objects. Parameters are addressed using 16-bit unsigned index number.

Submodule parameter can be made accessible for IO Controller by linking it to Record Data object.

4.7.1 Modules

Module represents process interface. Initial project template for PROFINET comes with three example modules already defined. They can be changed or removed depending on user preference.

The following section describes the different module properties users can set.

Name

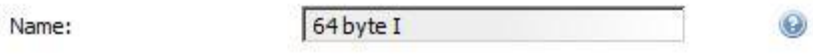


Figure 40: Module name input field

Human-readable name of the module. Maximum 100 characters. Should be concise, yet descriptive. Uniqueness is not enforced, but should be avoided as potential source of confusion.

Name of module is shown as its label in tree view. It will be put into software model code as well as device description code.

Identification number

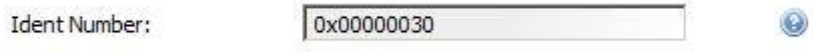


Figure 41: Module number input field

Identification number of the module. Unsigned 32-bit integer. Each module in device must have unique number. Numbers below 3 are reserved. Module number together with the name are displayed as Module's label, usually in both hexadecimal and decimal (in parenthesis) format e.g "Module: nineteen 0x13(19)".

When changing number value, one can use his/her preferred format: decimal, octal (with 0 prefix) or hexadecimal (with 0x prefix). New value won't be accepted when previously mentioned constraints are not met - the red error sign will appear next to input field.

Description

Figure 42: Module description input field

Additional text information about Module. It will be put into GSDML file.

Slot selection

Figure 43: Module Slot selection field

Specifies into which slot module is plugged. This selection is optional - if no Slot is selected the form will say "No Slot selected".

The user can select Slot for Module from existing ones, by using the *Browse* button (Figure 43 - the left one). Clicking the button will open simple list dialog with possible selection options. To plug Module into Slot, select its entry on the dialog's list and click *OK*. To unplug Module, select *No Slot* option. As Module can be plugged only in one Slot at the same time, only one option can be selected. To avoid being redundant, current selection is not available to choose from.

As already mentioned, no two Modules can be put into the same Slot simultaneously. Therefore, assigning Module to Slot X, will result in unplugging Module that was plugged to Slot X previously. The tool detects and warns user if this is about to happen - see Figure 44.

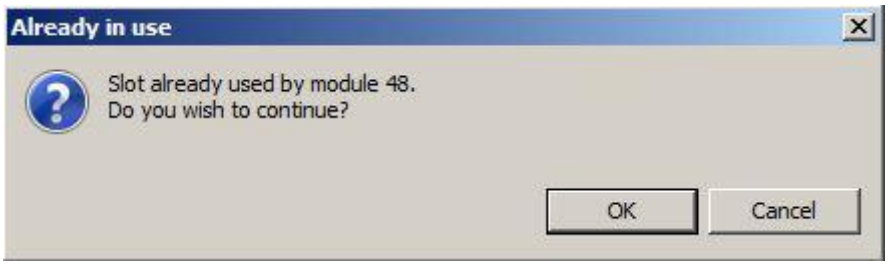


Figure 44: Warning if slot is already in use

The same effect of plugging Module into already existing Slot, can also be achieved by using Create Connection tool ( ) on model blocks.

The *Create* button - the right one on Figure 4, is a shortcut that allows the definition of a new Slot and assigns it to the Module at the same time.

4.7.2 Submodules

Submodule is a child object for Module. Each Module must have at least one Submodule defined. One can browse Module's children using tree view inside editor.

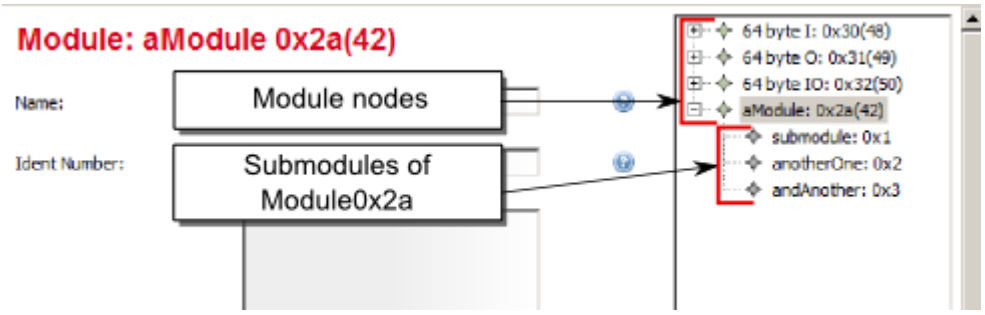


Figure 45: Fragment of editor for Module

When any Module is opened in editor, user is presented with tree representation of Module structure on the right side of editor tab - see Figure 45. Submodule page can be opened, simply by clicking associated tree node.

Figure 46: Module's submodule management field

Submodule is a child object for Module. From Module's editor page, user is able to create new Submodules and remove existing ones.

Use *Add Submodule* button (the left one on Figure 46) to create new Submodule and add it as child of current Module. Clicking on button will launch Create Submodule Wizard.

Use *Remove Submodules* button (the right one on Figure 46) to remove Module's Submodules. Clicking on button will open simple selection dialog with the list of Module Submodules. You can select more Submodules than one by having `Shift` key pressed while clicking on the entries.

Remember: Every Module must have at least one Submodule. The tool will not allow to remove all Submodules of the Module.

The following section gives an overview about the submodule properties the user can set.

4.7.2.1 Name

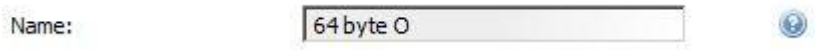


Figure 47: Submodule name input

Human-readable name of Submodule. Limited to 100 characters. Should be concise, yet descriptive. Uniqueness is not enforced, but should be avoided as potential source of confusion.

Name will be put both to software model file and to device description file.

4.7.2.2 Identification Number

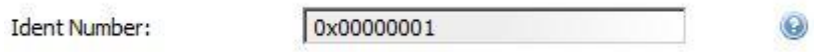


Figure 48: Submodule number input

Identifier of the Submodule. Must be unique within Module. Unsigned 32-bit integer number. Submodule number together with the name are displayed as Submodule's label, usually in both hexadecimal and decimal (in parenthesis) format e.g "Submodule: twelve 0xc(12)".

When changing number value, one can use his/her preferred format: decimal, octal (with 0 prefix) or hexadecimal (with 0x prefix). New value won't be accepted when previously mentioned constraints are not met - the red error sign will appear next to input field.

4.7.2.3 Description

Figure 49: Submodule description input

Additional text information about Submodule.

4.7.2.4 Data Direction

Figure 50: Submodule date direction input

Specifies a kind of interface the Submodule represents. It can be input, output, both or none. Should be consistent with input and output data type settings.

4.7.2.5 Input Data Type

Figure 51: Submodule input data type

Specifies type of input data. Possible options are:

- None - for when data direction is neither an Input nor Input/Output;
- Unsigned 8 - unsigned 8 bit integer number; values from 0 to 255; takes one byte;
- Unsigned 16 - unsigned 16 bit integer number; values from 0 to 65535; takes two bytes;
- Unsigned 32 - unsigned 32 bit integer number; values from 0 to 4294967295; takes four bytes;
- Signed 8 - signed 8 bit integer number; values from -128 to 127; takes one byte;
- Signed 16 - signed 16 bit integer number; values from -32768 to 32767; takes two bytes;
- Signed 32 - signed 32 bit integer number; values from -2147483648 to 2147483647; takes four bytes;
- Octet String - string of characters, each of them taking one byte;

Data type setting determines input unit. Input data length determines how many input units there are.

4.7.2.6 Input Data Length

Figure 52: Submodule input data length

Specifies length of input data in bytes. The value is a 16 bit unsigned integer. Should be consistent with data type. That means it has to be divisible by byte-size of the unit. Depending on data type, valid values are:

- None - 0;
- Unsigned 8 - 1, 2, 3, 4, ..., 65535;
- Signed 8 - 1, 2, 3, 4, ..., 65535;
- Octet String - 1, 2, 3, 4, ..., 65535;
- Unsigned 16 - 2, 4, 6, 8, ..., 65534;
- Signed 16 - 2, 4, 6, 8, ..., 65534;
- Unsigned 32 - 4, 8, 12, 16, ..., 65532;
- Signed 32 - 4, 8, 12, 16, ..., 65532;

4.7.2.7 Output Data Type

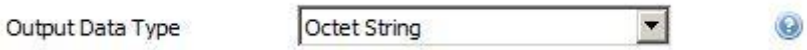


Figure 53: Submodule output data type

Specifies type of output data. Possible options are:

- None - for when data direction is neither an Output nor Input/Output;
- Unsigned 8 - unsigned 8 bit integer number; values from 0 to 255; takes one byte;
- Unsigned 16 - unsigned 16 bit integer number; values from 0 to 65535; takes two bytes;
- Unsigned 32 - unsigned 32 bit integer number; values from 0 to 4294967295; takes four bytes;
- Signed 8 - signed 8 bit integer number; values from -128 to 127; takes one byte;
- Signed 16 - signed 16 bit integer number; values from -32768 to 32767; takes two bytes;
- Signed 32 - signed 32 bit integer number; values from -2147483648 to 2147483647; takes four bytes;
- Octet String - string of characters, each of them taking one byte;

Data type setting determines output unit. Output data length determines how many output units there are.

4.7.2.8 Output Data Length

Figure 54: Submodule output data length

Specifies length of output data in bytes. The value is a 16 bit unsigned integer. Should be consistent with data type. That means it has to be divisible by byte-size of the unit. Depending on data type, valid values are:

- None - 0;
- Unsigned 8 - 1, 2, 3, 4, ..., 65535;
- Signed 8 - 1, 2, 3, 4, ..., 65535;
- Octet String - 1, 2, 3, 4, ..., 65535;
- Unsigned 16 - 2, 4, 6, 8, ..., 65534;

- Signed 16 - 2, 4, 6, 8, ..., 65534;
- Unsigned 32 - 4, 8, 12, 16, ..., 65532;
- Signed 32 - 4, 8, 12, 16, ..., 65532;

4.7.2.9 Subslot Selection

Figure 55: Submodule Subslot input

Just as Module can be plugged in the software model to Slot, Submodule can be plugged into Subslot. This selection is optional - if no subslot is selected, the form will say "No Subslot selected".

The user can select Subslot for Submodule from existing ones, by using the *Browse* button (Figure 55 - the left one). This can be only done, when parent Module is plugged into Slot. Otherwise, information dialog will pop-up - see Figure 56. To remedy this situation, plug Module into some Slot.

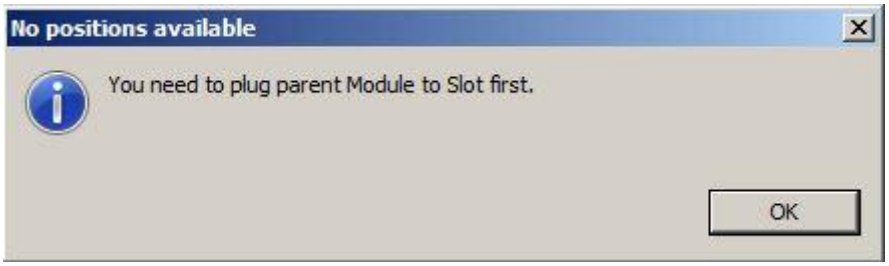


Figure 56: Message dialog when selection Subslot fails

When parent module is plugged, *Browse* button will open simple list dialog with possible selection options. These are Subslots of Slot, parent Module is plugged into. To plug Submodule into Subslot, select its entry on the dialog's list and click *OK*. To unplug Submodule, select *No Subslot* option. As Submodule can be plugged only in one Subslot at the same time, only one option can be selected. To avoid being redundant, current selection is not available to choose from.

As already mentioned, no two Submodules can be put into the same Subslot simultaneously. Therefore, assigning Submodule to Subslot X, will result in unplugging Submodule that was plugged to Subslot X previously. The tool detects and warns user if this is about to happen - see Figure 57.

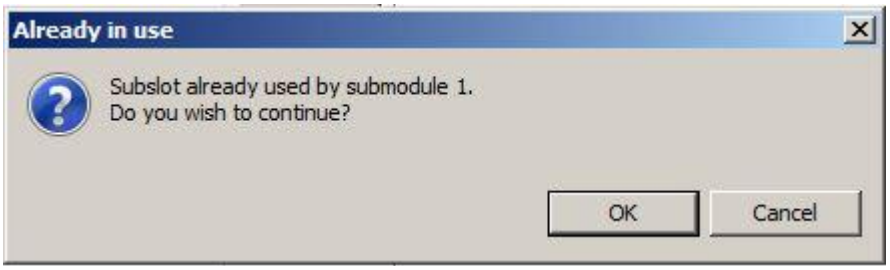


Figure 57: Dialog warnung that subslot is already in use

The *Create* button - the right one on Figure 55, is a shortcut that allows to define a new Subslot and assign it to the Submodule at the same time.

4.7.2.10 Parameters

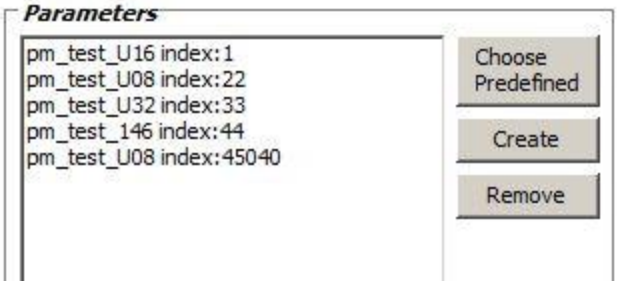


Figure 58: Submodule parameters input

User can supply Submodule definition with number of Parameters. Parameters that are currently added to Submodule are visible as entries in the list (Figure 58). Each entry is labeled with parameter name and index.

Parameter objects can be shared, so there is no need to create multiple instances of the same parameter. To add already existing parameter to Submodule parameters, use *Choose Predefined* button. Clicking the button will open simple selection dialog with list of Parameters available for selection. Please note, that parameters that are already assigned to the Submodule, are not available for selection.

To add predefined parameters to the Submodule, select their entries in the dialog and click *OK*. You can select multiple rows by holding `Shift` key while selecting.

User can also add new parameter to Submodule. Clicking on *Create* button will launch Create Parameter Wizard. After wizard finishes, the newly defined Parameter will be added to the list. It will then, also be available to add for other Submodules.

To remove object from list of Submodule Parameters, use *Remove* button. First, using mouse and `Shift` key highlight all entries on the Parameters list that you wish to remove. Then, click the button. Notice, that this only removes Submodule to Parameter reference, Parameter will still remain in project and other Submodules which use it, will remain unaffected. However, if you really want to remove Parameter object all-together check Parameters chapter.

4.7.3 Create a new Module

User can have much more Modules in his/her Software Model than the three ones predefined in new project template. One can add new Module to the model by selecting *Modules* editor block and using Create Object tool ( ).

New object is defined using Wizard dialog. The wizard can have up to two pages: first one to specify Module properties and optional second one to select Slot.

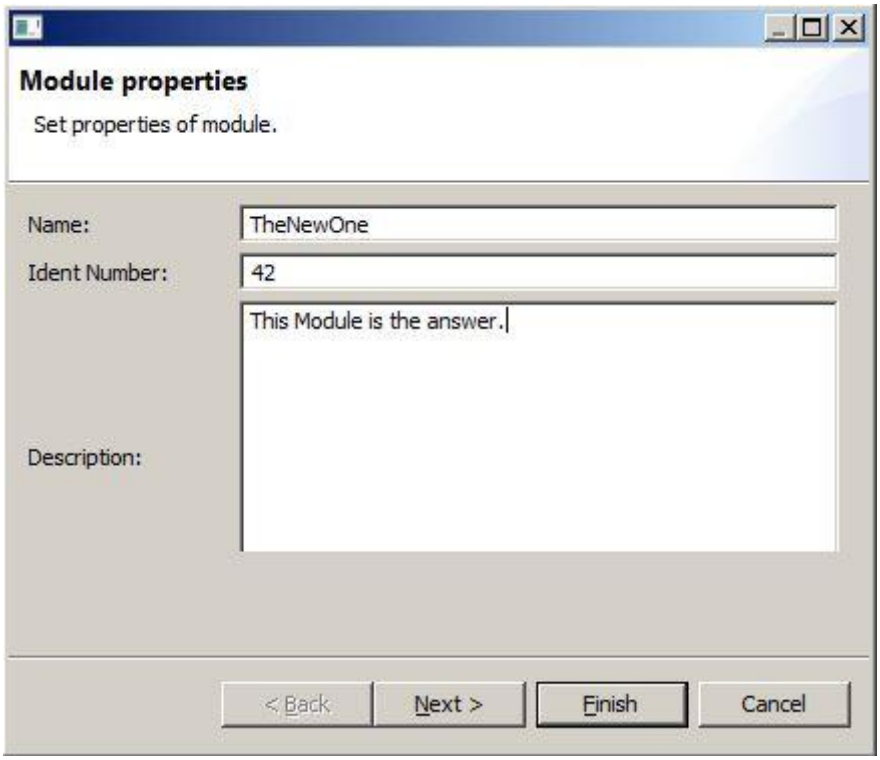


Figure 59: Create Module Wizard

First page of the wizard is presented on Figure 59. Completing this page is needed to create the Module. The properties: name, ident. number and description are equivalent to these described in chapter 4.7.1. You have to at least provide valid name and address for the wizard to continue. Remember, that Module number must be unique.

After name and identification number are specified, the user is free to finish the wizard by clicking the *Finish* button. Optionally, he can proceed to the next page where Module can be assigned to existing Slot. The page is presented on Figure 60.

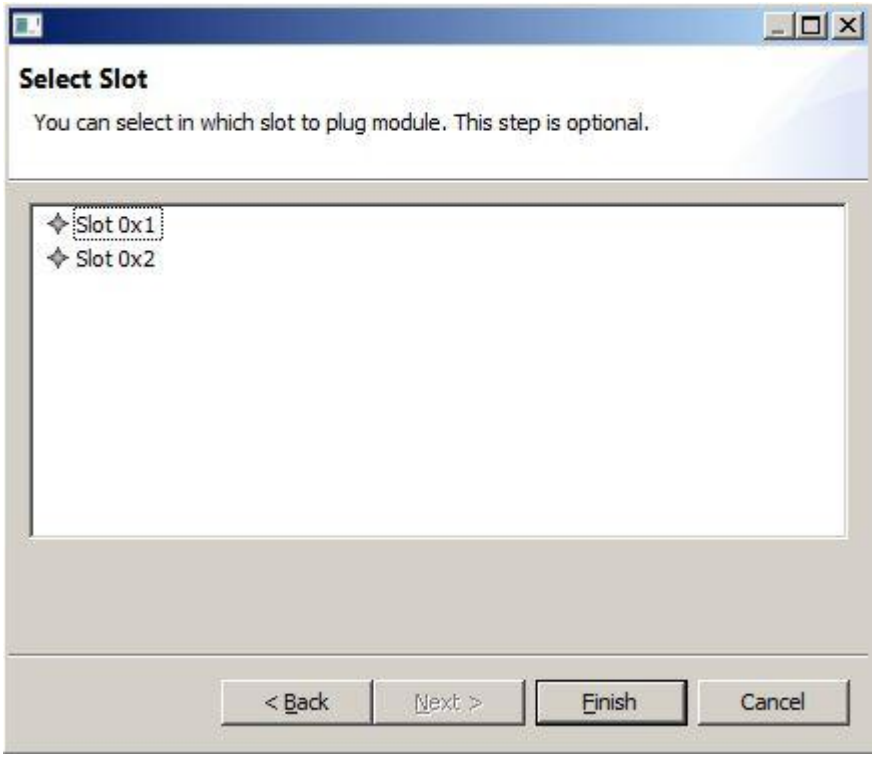


Figure 60: Create Module Wizard: Slot selection

Selecting Slot that is already used will result in warning message appearing on top of wizard page. If the user will decide to proceed with that selection, previous Slot assignment will be overridden.

4.7.4 Create a new Submodule

Submodules are fully contained in their parent Modules. Therefore, one has to use Module definition to create and remove them.

User is assisted through Submodule creation with New-Submodule-Wizard. The wizard has up to two pages. The first one allows the specification of basic properties and the second one allows the Subslot assignment. The second page is only displayed if Submodule's parent Module is plugged into Slot.

First page is shown on Figure 61. The attributes to specify are the same as described in the chapters 4.7.2.1-4.7.2.8. Minimal input required to proceed is to provide valid name and identification number for the new Submodule. Wizard automatically suggests the name to be the same as parent Module's, but user can change it for more descriptive one. Please remember that identification number has to be unique within Module to be accepted.

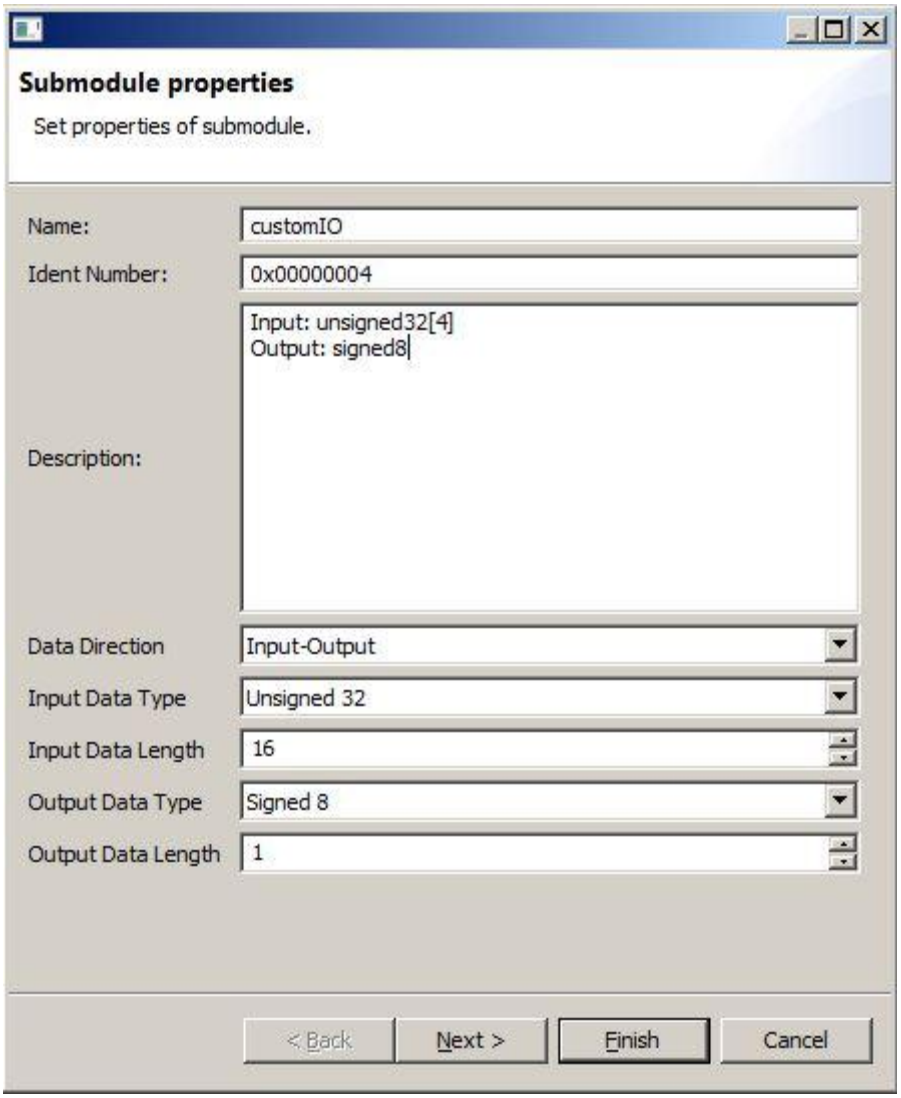


Figure 61: New Submodule Wizard

The optional second page is shown on Figure 62. If Submodule's parent is plugged into Slot, user can plug new Submodule into Subslot, by selecting an entry on the list. This step can be skipped.

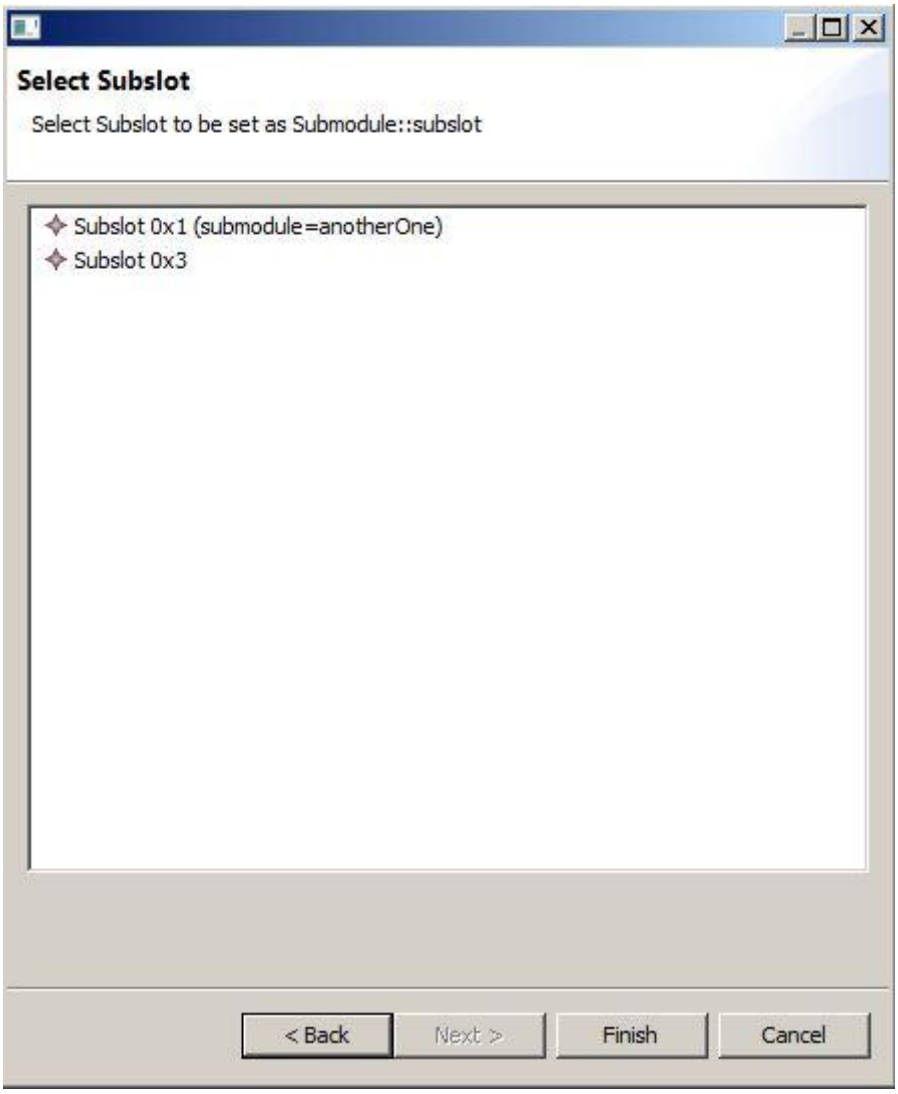


Figure 62: New Submodule Wizard: Subslot selection

The wizard will warn if selected Subslot is already used by other Submodule. Because only one Submodule can be plugged into Subslot at the same time, the previous assignment would have to be removed.

4.7.5 Slots

Slots are used to represent addresses within the device. In the initial PROFINET project, there are two example Slots defined. They can be modified or removed by the user.

The following section describes the different properties a user can set for slots.

Number

Figure 63: Slot number input

Identification number of the Slot. Must be unique. Value is a 16 bit unsigned integer. Value "0" is reserved as Device Access Point (DAP).

Slot's number is displayed as its label, usually in both hexadecimal and decimal (in parenthesis) format e.g "Slot 0x10(16)".

When changing number value, one can use his/her preferred format: decimal, octal (with 0 prefix) or hexadecimal (with 0x prefix). New value won't be accepted when previously mentioned constraints are not met - the red error sign will appear next to input field.

Description

Figure 64: Slot description input

Optional user text. Can be used to put comments about Slot.

Module Selection

Figure 65: Slot Module input

Specifies which Module (if any) is plugged into this Slot. If no Module is selected, the form will say "No Module selected".

The user can select Module for Slot from existing ones, by using the *Browse* button (Figure 65 - the left one). Clicking the button will open simple list dialog with possible selection options. To plug Module into Slot, select its entry on the dialog's list and click *OK*. To unplug Module, select *No Module* option. Only one Module can be plugged into Slot, so only one selection is possible. To avoid being redundant, current selection is not available to choose from.

As already mentioned, no two Modules can be put into the same Slot simultaneously. Therefore, assigning Module to Slot X, will result in unplugging Module that was plugged to Slot X previously. The tool detects and warns user if this is about to happen - see Figure 66.

Figure 66: Module already in use warning

The same effect of plugging Module into Slot, can also be achieved by using Create Connection tool ( ) on model blocks.

The *Create* button - the right one on Figure 65, is a shortcut that allows to define new Module and plug it to the Slot at the same time.

4.7.6 Subslots

Subslot is a child object for Slot. Each Slot must have at least one Subslot defined. One can browse Slot's children using tree view inside editor.

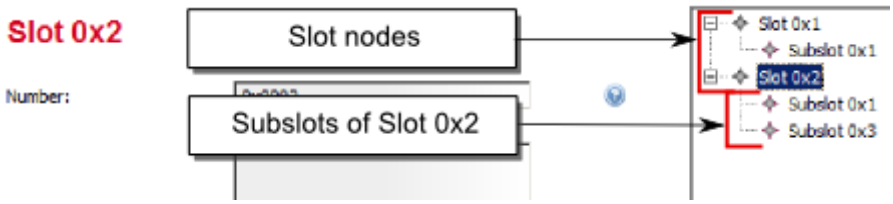


Figure 67: Fragment of editor page for Slots

Subslot is a child object for Slot. From Slot's editor page, user is able to create new Subslots and remove existing ones.

Figure 68: Subslots management input

Use *Add Subslot* button (the left one on Figure 68) to create new Subslot and add it as child of current Slot. Clicking on button will launch Create Subslot Wizard.

Use *Remove Submodules* button (the right one on Figure 68) to remove Slot's Subslots. Clicking on button will open simple selection dialog with list. You can select more Subslots than one by having `Shift` key pressed while clicking on the entries.

Remember: Every Slot must have at least one Subslot. The tool will not allow to remove all Subslots of a Slot.

The following section describes the different properties a user can set for subslots.

Number

Figure 69: Subslot number input

Identification number of Subslot. Must be unique within Slot. Value is a 16 bit unsigned integer.

Subslot's number is displayed as its label, usually in both hexadecimal and decimal (in parenthesis) format e.g "Subslot 0x11(17)".

When changing number value, one can use his/her preferred format: decimal, octal (with 0 prefix) or hexadecimal (with 0x prefix). New value won't be accepted when previously mentioned constraints are not met - the red error sign will appear next to input field.

Description

Figure 70: Subslot description input

Optional user text. Can be used to write comments about Subslot.

Submodule Selection

Figure 71: Subslot Submodule input

Just as Module can be plugged in the software model to Slot, Submodule can be plugged into Subslot. This selection is optional - if no Submodule is selected, the form will say "No Submodule selected".

User can select Submodule for Subslot from existing ones, by using the *Browse* button (Figure 71 - the left one). This can be only done, when parent Slot has Module assigned. Otherwise, information dialog will pop-up - see Figure 72. To remedy this situation, plug some Module into parent Slot.

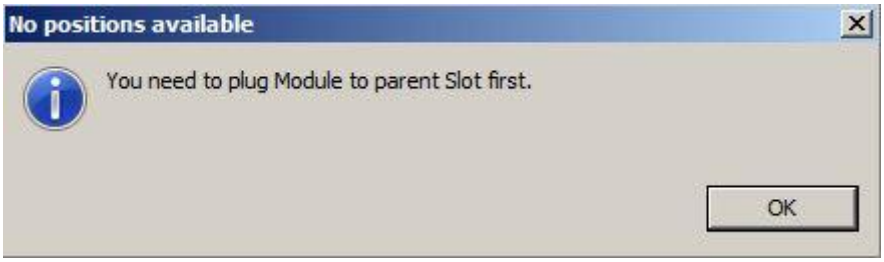


Figure 72: Message dialog when selecting Submodule fails

When parent Slot has Module plugged, *Browse* will open simple list dialog with possible selection options. These are Submodules of parent Slot's Module. To plug Submodule into Subslot, select its entry on the dialog's list and click *OK*. To unplug Submodule, select *No Submodule* option. As Submodule can be plugged only in one Subslot at the same time, only one option can be selected. To avoid being redundant, current selection is not available to choose from.

As already mentioned, no two Submodules can be put into the same Subslot simultaneously. Therefore, assigning Submodule to Subslot X, will result in unplugging Submodule that was plugged to Subslot X previously. The tool detects and warns user if this is about to happen - see Figure 73.

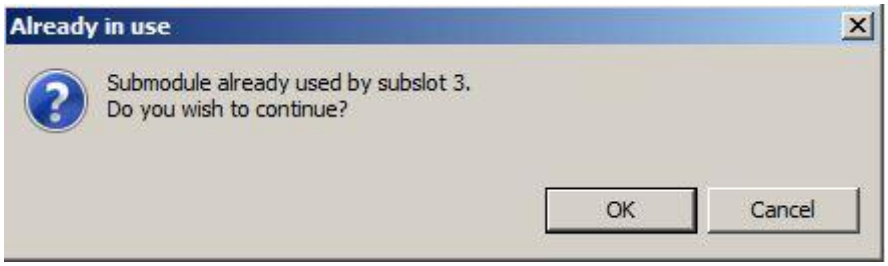


Figure 73: Warning when submodule is alread in use

The *Create* button - the right one on Figure 71, is a shortcut that allows to define new Submodule and assign it to the Subslot.

Records

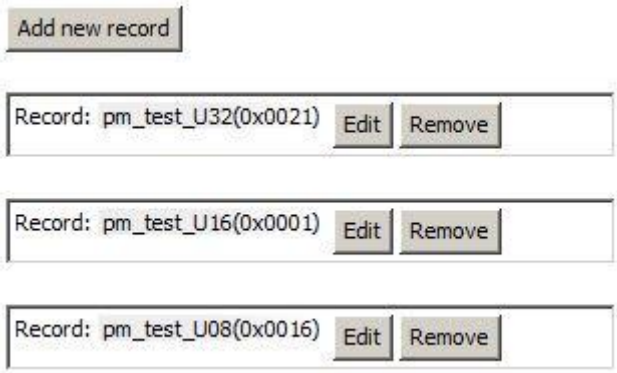


Figure 74: Subslot records input

Subslot can be a parent for number of Record Data Objects. The Record definition is made of two elements: Parameter and value.

On Subslot editor page, there is an *Add New Record* button (top of Figure 74). Below each Subslot Record is listed (if any are defined). Record entry is labeled with Parameter name and index.

To create new Record, click *Add New Record* button. It will open two-page wizard. First page, shown on Figure 75 is for selecting Parameter. User can choose from any previously defined Parameters.

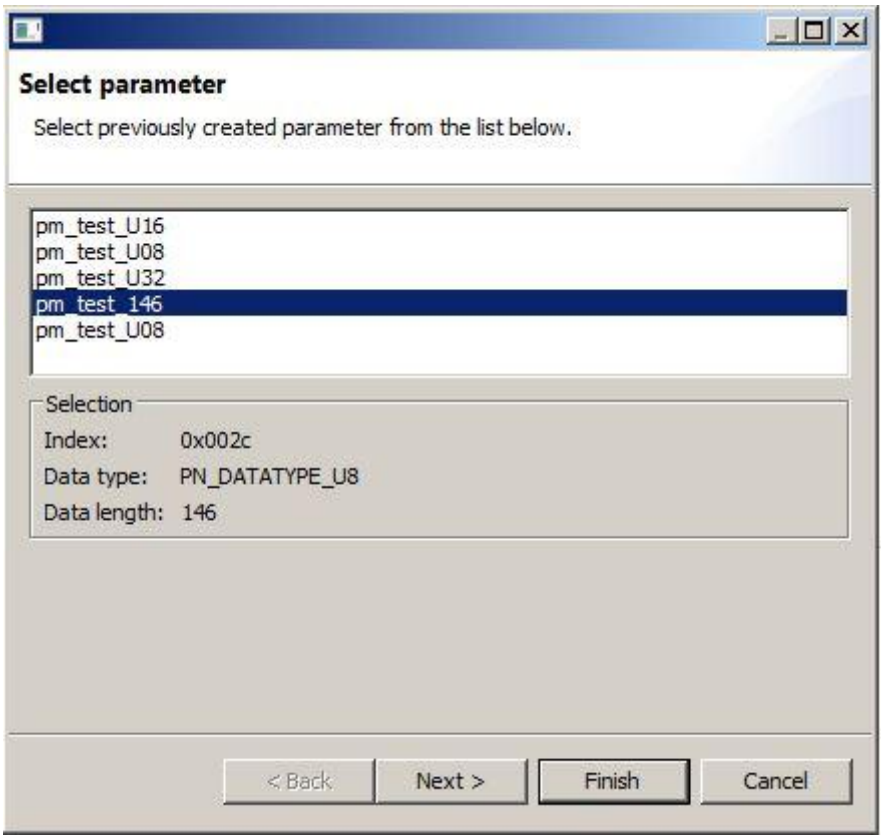


Figure 75: Create new Record Wizard

On the second page, visible on Figure 76, user can specify value of data Record. Value is dependent on data type and data length of the Parameter. Numeric values are automatically initialized to zero, and have to be specified as sequence separated by whitespaces. The Wizard will check if input conforms to data length and data type, and display error messages if not.

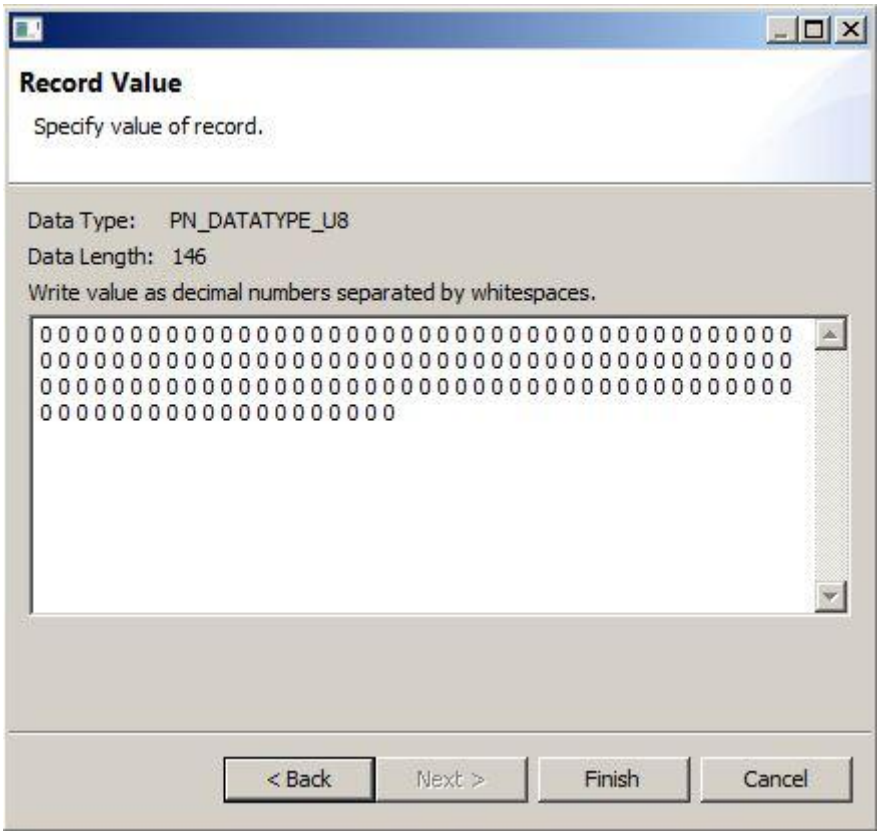


Figure 76: Create new Record Wizard: Record Value

After Record is defined it can be modified by using *Edit* button, next to its label (see Figure 74). Clicking on the button will re-launch wizard for selected Record.

Record can be removed from the model by using *Remove* button, next to its label (see Figure 74).

4.7.7 Create a new Slot

User can have much more Slots in his/her Software Model than the two ones predefined in new project template. One can add new Slot to the model by selecting *Slots* editor block and using Create Object tool ( ).

New object is defined using Wizard dialog. The wizard can have up to two pages: first one to specify Slot properties and optional second one to select Module.

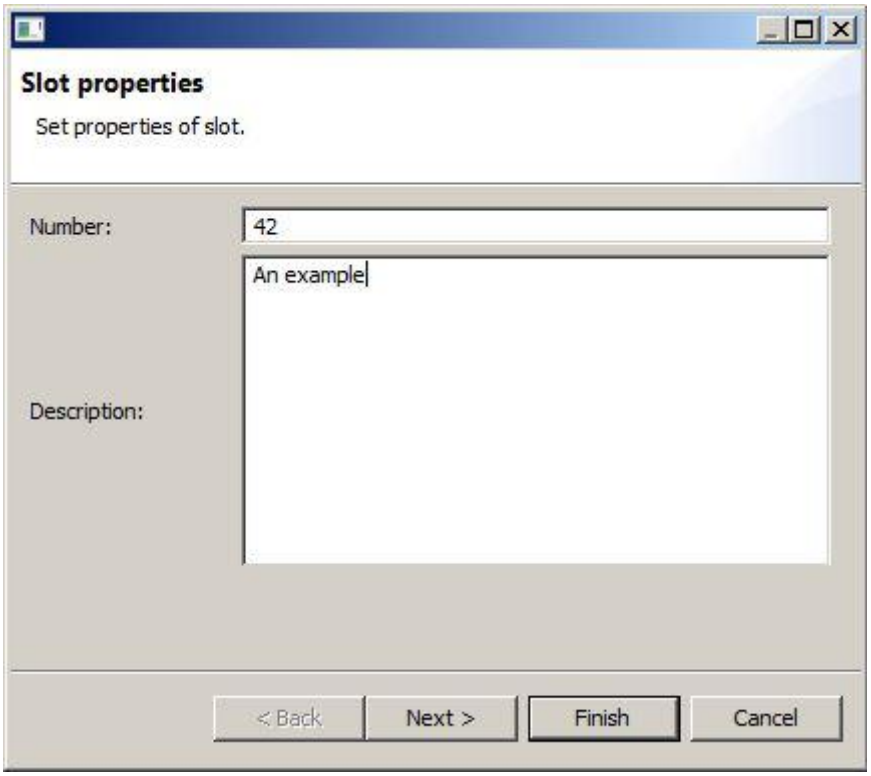


Figure 77: Create Slow Wizard

First page of the wizard is presented on Figure 77. Completing this page is needed to create the Slot. User has to at least provide valid Slot number to continue. Remember that Slot number must be unique.

After number is specified, the user is free to finish the wizard by clicking the *Finish* button. Optionally, he can proceed to the next page where Module can be assigned to Slot. The page is presented on Figure 78.

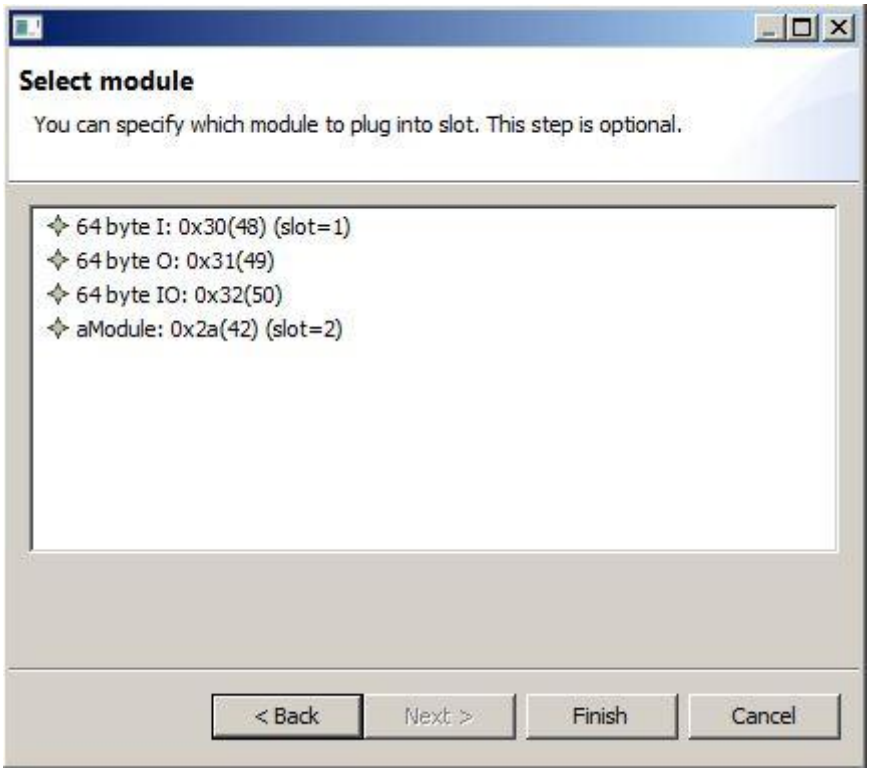

Select module
You can specify which module to plug into slot. This step is optional.
64 byte I: 0x30(48) (slot=1)
64 byte O: 0x31(49)
64 byte IO: 0x32(50)
aModule: 0x2a(42) (slot=2)
< Back
Next >
Finish
Cancel

Figure 78: Create Slot Wizard: Module selection

Selecting Module that is already used will result in warning message appearing. If the user will decide to proceed with that selection, previous Slot assignment will be overridden.

4.7.8 Create a new Subslot

Subslots are fully contained in their parent Slots. Therefore, one has to use Slot definition to create and remove them.

User is assisted through Subslot creation with New-Subslot-Wizard. The wizard has up to two pages. First one, for specifying basic properties and second one, for Submodule assignment. The second page is only displayed if Subslot's parent has Module assigned.

First page is shown on Figure 79. User has to specify number for Subslot. Please remember that number has to be unique within Slot to be accepted.

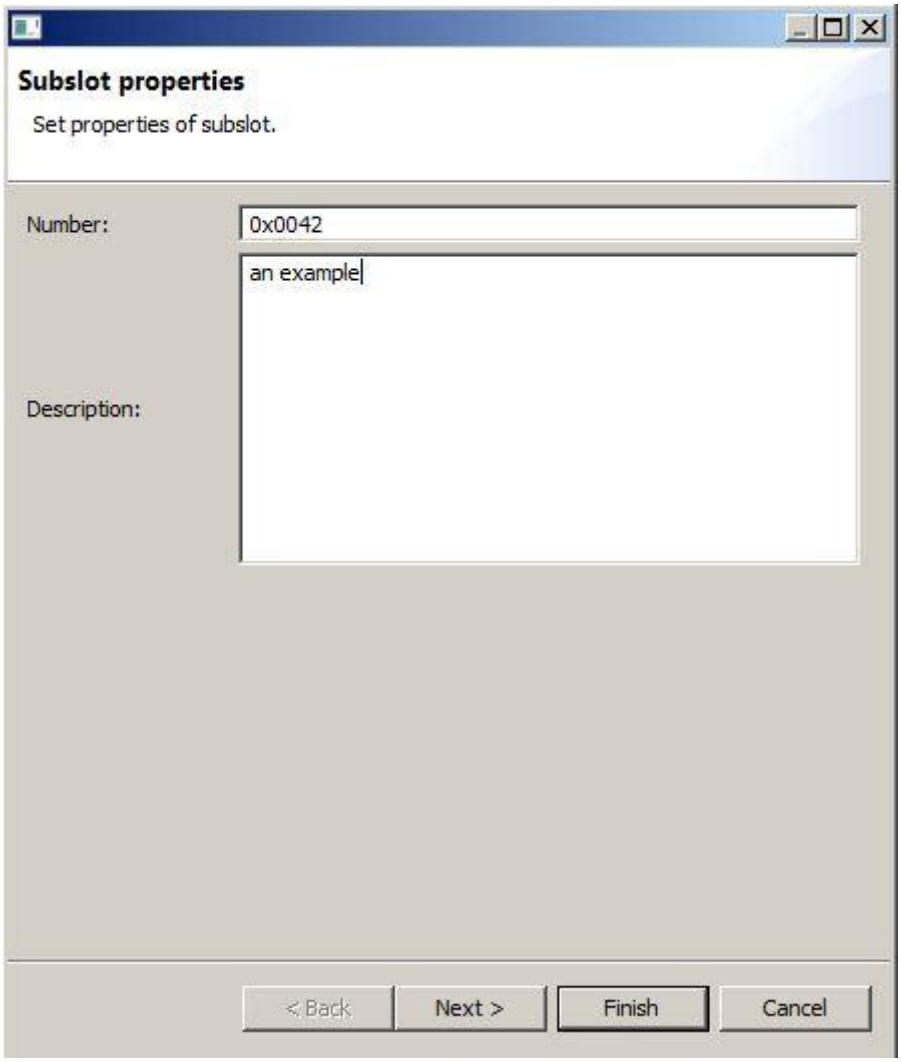


Figure 79: New Subslot Wizard

The optional second page is shown on Figure 80. If Subslot's parent has Module assigned, user can plug Submodule into new Subslot. This step can be skipped.

Figure 80: New Subslot Wizard: Submodule selection

The wizard will warn if selected Submodule is already used by other Subslot. Because only one Submodule can be plugged into Subslot at the same time, the previous assignment would have to be removed.

4.7.9 Parameters

Parameters are software model objects that can be used to parameterize Submodules and Record Data objects. List of all Parameters currently present in software model is available under *Parameters* editor block.

The following section gives an overview about settings for parameters.

Name

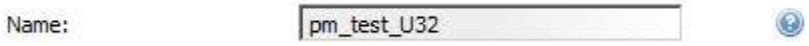


Figure 81: Paramter name input

Human-readable name of the Parameter. Maximum number of characters is 100.

Index

Figure 82: Parameter index input

Address of data represented by Parameter. Value is a 16 bit unsigned number.

Description

Figure 83: Parameter description input

Optional user text. Can be used to add comments about Parameter.

Data Type

Figure 84: Parameter data type input

Specifies data type of the Parameter. Possible options are:

- None - no data;
- Unsigned 8 - unsigned 8 bit integer number; values from 0 to 255; takes one byte;
- Unsigned 16 - unsigned 16 bit integer number; values from 0 to 65535; takes two bytes;
- Unsigned 32 - unsigned 32 bit integer number; values from 0 to 4294967295; takes four bytes;
- Signed 8 - signed 8 bit integer number; values from -128 to 127; takes one byte;
- Signed 16 - signed 16 bit integer number; values from -32768 to 32767; takes two bytes;
- Signed 32 - signed 32 bit integer number; values from -2147483648 to 2147483647; takes four bytes;
- Octet String - string of characters, each of them taking one byte;

Data type setting determines data unit. Data length determines how many units there are in the data.

Data Length

Figure 85: Paramater data length input

Specifies length of data in bytes. The value is a 16 bit unsigned integer. Should be consistent with data type. That means it has to be divisible by byte-size of the unit. Depending on data type, valid values are:

- None - 0;
- Unsigned 8 - 1, 2, 3, 4, ..., 65535;
- Signed 8 - 1, 2, 3, 4, ..., 65535;
- Octet String - 1, 2, 3, 4, ..., 65535;
- Unsigned 16 - 2, 4, 6, 8, ..., 65534;
- Signed 16 - 2, 4, 6, 8, ..., 65534;
- Unsigned 32 - 4, 8, 12, 16, ..., 65532;
- Signed 32 - 4, 8, 12, 16, ..., 65532;

4.7.10 Create/Delete Parameters

New parameters can be created (and associated with Submodules) using Submodule editor page. Parameter creation is done with assistance of wizard.

Create New Parameter Wizard has only one page, shown on Figure 86. Page is used to specify Parameter properties. User has to at least provide valid name of new Parameter.

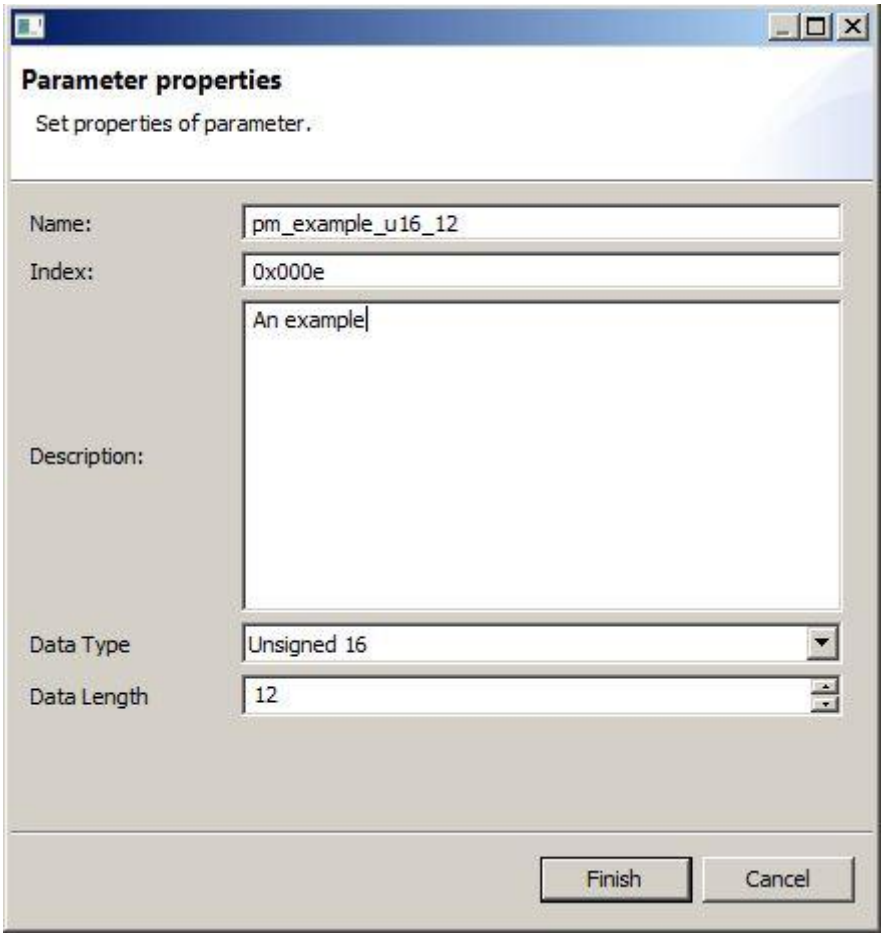


Figure 86: Create new Parameter Wizard

Parameter objects that are no longer used in any Submodules or Record Data objects, can be deleted from software model. This can be done either through editor or Outline.

On editor page for Parameters, user can select unwanted Parameter nodes and click `Delete` key. In Outline, Parameters can be removed using context menu - see Figure 87.

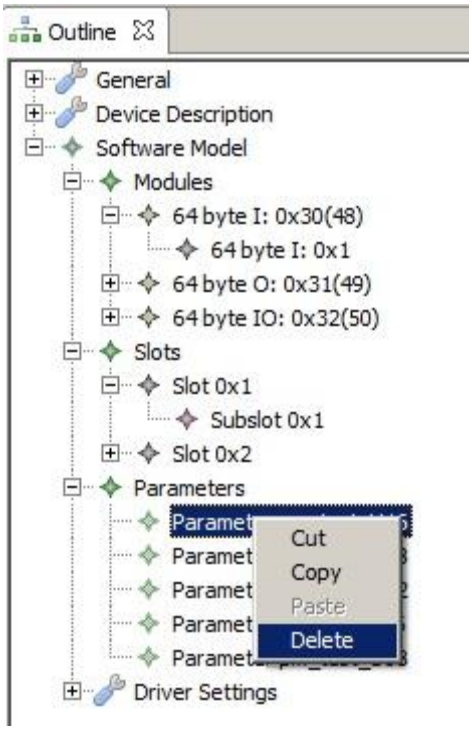


Figure 87: Deleting Parameter using Outline View

4.7.11 Settings reference

A detailed reference of all available settings for port PROFINET IO stack can be found in the online help of the DesignTool. Please open the help system via menu “Help” → “Help contents”. Then select “port PROFINET Tool User Guide” → “Settings Reference”.

5 Appendix A — Abbreviations

IDE	Integrated Development Environment
PNIO	PROFINET I/O

6 Appendix B – Common Problems

Global scope

Warning: Setting not found in source, default value will be used instead.
Solution: This warning appears during project file loading, if setting that is expected for given project type could not be found in project file. This can be a normal effect of application update that will not appear again after project is saved.

Error: Error occurred during licence decryption.
Solution: This error means that application could not read licence file contents and in result will run in limited (Demo) mode. If you own full licence for *port DesignTool* product, check location of your licence file. If problem persists, contact *port* user support.

Error: I/O error occurred while loading licence file.
Solution: This error means that application could not read licence file contents and in result will run in limited (Demo) mode. If you own full licence for *port DesignTool* product, check location of your licence file. If problem persists, contact *port* user support.

Error: Update failed
Example: Update failed: invalid location of repository
Solution: Error occurs when requested auto-update operation fails. Check your Internet connection. If problem persists, you may try to update product manually - by downloading new version from *port* home page.

Info: Running as demo version.
Solution: This report informs that application runs in limited, instead of full, version. If you don't own licence for full version of *port Design Tool*, you may ignore this info and continue with product evaluation or consider purchasing full licence. If you already own full licence, check if licence file is installed correctly. If problem persists, contact *port* user support.

Project scope

Error: Unable to read from file.
Examples: Unable to read from file: /home/user/project_B.dsntool (The system cannot find the file specified)
Unable to read from file: C:\Documents\user\project_A.dsntool (Content is not allowed in the prolog)
Unable to read from file: C:\Documents\user\proj_1.dsntool: Project type is not supported
Solution: This error appears when project file cannot be read. You should check if file exist in loca-

tion specified and if user running application has read access to the file. It is also possible that file is corrupted or the tool doesn't support this project type. In the case of the latter, you may need to download new feature to your application.

Error: Unable to load data from project.
Solution: This error appears when editor is initialized for project that cannot be read. Check instructions for "Unable to read from file" problem.

Error: Error while project save.
Example: Error while project save: C:\Documents\user\project_1.dsntool (Access denied)
Solution: This error appears when project file cannot be written. You should check if user running application has write access to the file and whether there is free space available on device.

Error: Unable to write to file...
Example: Unable to write to file: C:\Documents\user\file.c Access denied
Solution: This error appears when one of project-related files cannot be written. You should check if user running application has write access to the file and whether there is free space available on device.

Error: Unable to get setting value
Solution: This error usually appears during code generation. This indicates that the project file was corrupted or is opened with wrong (e.g. obsolete) version of the tool. Most likely, the direct reason is setting value written in incorrect format. It should be possible to remedy this, by changing value of the setting that was identified as source of problem.

Warning: Setting value not found. Default value (XXX) used instead.
Solution: This warning may appear during code generation. It indicates that value of configuration setting was not specified in project. The warning will cease to appear when user will set the value and save project. The warning may be an indication that user haven't finished all project tasks.

Warning: Loaded project has version higher than structure supported by tool.
Example: Loaded project has version higher (1.2) than structure supported by tool (1.0).
Solution: This warning appears when project file is opened with obsolete version of the tool. It is recommended to update tool before modifying project or to create backup of project file.

7 Appendix C – PROFINET Task Reference

UNFINISHED - TASKS NEED REVIEW

Task: Configure device MAC address.
Location: Network.
Details: In case of some devices, user must specify device's MAC address explicitly. Change the address to something other than 00:00:00:00:00:00 to silence the task.

Task: Supply device information.
Location: Device Identity.

Details: GSDML file should contain short textual device description. Task will be open if Device Information setting is empty.

Task: Set your PI-assigned Vendor ID.
Location: Device Identity.
Details: As a part of device description, Vendor ID is specified. This 16-bit integer number is assigned by PROFINET International. For new projects, Vendor ID setting is initialized with *port'*sown number (0x28c). You need to change this value for the task to be completed.

Task: Specify description for every module.
Location: Modules.
Details: For device description, each defined device module can contain short information text. This task checks presence of these texts, the lack of whose may suggest that the Module is not configured yet.

Task: Configure GSDML attributes: name of manufacturer and device
Location: GSDML.
Details: According to the GSDML specification, name of device description file has to be made of certain elements. These elements are configurable via GSDML configuration block. User should change these settings so they represent his use case.

8 Appendix D – Common PROFINET problems

Error: An exception occurred while delegating evaluation of the XXX constraint on YYY
Examples: An exception occurred while delegating evaluation of the 'ValueIsOptionsIndex' constraint (...)
An exception occurred while delegating evaluation of the 'IntValueBetweenLimits' constraint (...)
Solution: The error will be reported if setting value is in incorrect form. This most likely means that the project was loaded from corrupted file. In case of two given examples it should be possible to fix problem by changing values of problematic settings.

Warning: XXX constraint is not satisfied for YYY
Examples: 'IntSetting::IntValueBetweenLimits' constraint is not satisfied for (...)
'LimitedSetting::ValueIsOptionsIndex' constraint is not satisfied for (...)
Solution: The error will be reported if configuration setting or object violates project constraints. Since application is designed to keep user from breaking these constraints during normal project editing, presence of this report may mean that the project file was corrupted. Still, it should be possible to fix most of problems of this type, by simply editing reported configuration element. Meaning of some constraint violations is described below:

- LimitedSetting::ValueIsOptionsIndex - Setting value is incorrect; Can be fixed by changing setting value;
- IntSetting::IntValueBetweenLimits - Numeric value is not in valid range; Can be fixed by changing setting value;

- [Module, Slot, Submodule, Subslot]::AddressInRange - Identification number of software model object is not in valid range; Can be fixed by changing the number in question;
- [Module, Submodule, Parameter]::NameLength - name of object is longer than 100 characters; Can be fixed by changing the name;
- Submodule::[DataLengthInRange, DataLenghtOutRange] - data length attribute of Submodule object is invalid; Can be fixed by changing the value of problematic field;
- Parameter::IndexInRange - index attribute is not a 16-bit integer; Can be fixed by changing the value of index.

Error: Object has more children than specified in setting XXX
Solution: The error is reported if Software Model definition is inconsistent with Resource Management settings. The possible cases are:

- There is more Modules defined than Number of device Modules setting value;
- There is more Slots defined than Number of device Slots setting value;
- There is more Submodules in Module than Number of device Submodules per Module setting value;
- There are more Subslots in Slot than Number of device Subslots per Slot setting value.

These problems can be fixed by either removing excess objects or by correcting corresponding setting value.

Error: Unable to save new project.
Solution: The error is reported during new project creation, when save of initial project contents to file fails. This renders created project unable for future use. User should check if disk has enough free space and whether application has write permission for the file in question.

Error: IO error while saving GSDML file.
Solution: The error is reported when attempt to save GSDML file to disk fails. User should check if disk has enough free space and whether application has write permission for the file in question.

Error: Problem while saving GSDML stylesheet.
Solution: The error is reported when attempt to copy stylesheet file to the same location as GSDML file fails. User should check if disk has enough free space and whether application has write permission for the file in question.

Error: Code generation failed.
Example: Code generation failed. Record has no parameter.
Solution: Error is reported when due to incorrect project configuration generation of code was unable to finish. Error report should contain a link to object that caused the problem, and which needs to be fixed.